UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RAFAEL HENRIQUE TIBOLA

DETECÇÃO DE SPAM EM MENSAGENS SMS UTILIZANDO APRENDIZAGEM DE MÁQUINA

TRABALHO DE CONCLUSÃO DE CURSO

MEDIANEIRA

2019

RAFAEL HENRIQUE TIBOLA

DETECÇÃO DE SPAM EM MENSAGENS SMS UTILIZANDO APRENDIZAGEM DE MÁQUINA

Trabalho de Conclusão de Curso apresentado ao Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Bacharel em Computação".

Orientador: Prof. Dr. Arnaldo Candido Junior

MEDIANEIRA

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Coordenação do Curso de Ciência da Computação

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

DETECÇÃO DE SPAM EM MENSAGENS SMS UTILIZANDO APRENDIZAGEM DE MÁQUINA

Por

RAFAEL HENRIQUE TIBOLA

Este Trabalho de Conclusão de Curso foi apresentado às 13:00h do dia 19 de novembro de 2018 como requisito parcial para a obtenção do título de Bacharel no Curso de Ciência da Computação, da Universidade Tecnológica Federal do Paraná, Câmpus Medianeira. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Arnaldo Candido Junior
UTFPR - Câmpus Medianeira

Prof. Dr. Alan Gavioli
UTFPR - Câmpus Medianeira

Prof. Dr. Evando Carlos Pessini
UTFPR - Câmpus Medianeira

A folha de aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

TIBOLA, Rafael Henrique. DETECÇÃO DE SPAM EM MENSAGENS SMS UTILIZANDO APRENDIZAGEM DE MÁQUINA. 57 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2019.

O SMS (Short Message Service) é ainda um dos serviços de comunicação móvel mais simples e práticos para alcançar consumidores, sendo independente da conexão com uma rede de Internet ou da capacidade dos aparelhos. Algumas aplicações fornecem recursos para o envio das mensagens SMS, mas por estarem presentes na Internet, há espaço para que usuários maliciosos as utilizem para realizar envios de *spam*. No âmbito da Inteligência Artificial, áreas como a Aprendizagem de Máquina e o estudo de línguas podem se mostrar grandes aliados no desenvolvimento de sistemas que auxiliem a filtragem de mensagens *spam*. Neste trabalho, apresenta-se como foram alcançadas melhorias no desempenho do algoritmo de classificação Bayesiano Ingênuo, utilizando-o para classificar mensagens SMS apoiado por Redes Neurais Artificiais e vetores Word Embedding, utilizados para prever e generalizar probabilidades para palavras que não foram utilizadas no treinamento do classificador.

Palavras-chave: classificação, filtro de mensagens, inteligência artificial, Bayesiano

ABSTRACT

TIBOLA, Rafael Henrique. SPAM DETECTION IN SMS MESSAGES USING MACHINE LEARNING. 57 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2019.

SMS (Short Message Service) is still one of the simplest and most practical mobile communication services to reach consumers, regardless of the connection to an Internet network or the capacity of the handsets. Some applications provide resources for sending SMS messages, but because they are present on the Internet, there is room for malicious users to use them to send spam. In the field of Artificial Intelligence, areas such as Machine Learning and language studies may prove to be great allies in the development of systems that aid in the filtering of spam messages. In this study, it is shown how improvements in the performance of the Bayesian Naive classification algorithm were reached, using it to classify SMS messages supported by Artificial Neural Networks and Word Embedding vectors, used to predict and generalize probabilities for words that were not used in the training of the classifier.

Keywords: classification, messages filter, artificial intelligence, Bayesian

LISTA DE FIGURAS

FIGURA 1 – A Hierarquia do aprendizado. 13
FIGURA 2 – O *Perceptron* 16
FIGURA 3 – Função passo 17
FIGURA 4 – Função Sigmoide Logística 18
FIGURA 5 – Função tangente hiperbólica 18
FIGURA 6 – Problema linearmente separável 19
FIGURA 7 – Rede feedforward multicamadas 20
FIGURA 8 – Problema linearmente separável 20
FIGURA 9 – Representação One-Hot 23
FIGURA 10 – Modelo CBOW 25
FIGURA 11 – Skip-gram 26
FIGURA 12 – Medidas RSQ obtidas com o treinamento de 500 épocas para os modelos de RNA. 45
FIGURA 13 – Medidas RSQ obtidos com o treinamento de 100 épocas para os modelos de RNA. 47
FIGURA 14 – Medidas RSQ obtidas com o treinamento de 1000 épocas para os modelos de RNA. 48
FIGURA 15 – Medidas RSQ obtidos com o treinamento de 2.000 épocas para os modelos de RNA. 50
FIGURA 16 – Medidas de avaliação das RNAs sobre as mensagens do *corpus*. 53

LISTA DE TABELAS

TABELA 1 – Mensagens organizadas na estrutura de DataFrame. 33
TABELA 2 – Palavras e probabilidades armazenadas em $P_{palavras}$ 34
TABELA 3 – Palavras e seus embeddings 35
TABELA 4 – Experimentos com algoirtmo NaiveBayesMultinomialText 41
TABELA 5 – Avaliação dos resultados obtidos pelas probabilidades calculadas. 43
TABELA 6 – Medida RSQ obtidos com o treinamento de 500 épocas para os modelos de RNA. 43
TABELA 7 – Medida RSQ obtidos com o treinamento de 100 épocas para os modelos de RNA 44
TABELA 8 – Medida RSQ obtidos com o treinamento de 1000 épocas para os modelos de RNA. 46
TABELA 9 – Medida RSQ obtidos com o treinamento de 2.000 épocas para os modelos de RNA 49
TABELA 10 – Avaliação dos modelos de RNA para mensagens *spam*. 51
TABELA 11 – Avaliação dos modelos de RNA para mensagens *ham*. 51

LISTA DE SIGLAS

SMS	Short Message Service
API	Applications Programming Interface
spam	Sending and Posting Advertising in Mass
IA	Inteligência Artificial
AM	Aprendizagem de Máquina
PLN	Processamento da Língua Natural
RNA	Redes Neurais Artificiais
CPU	Central Processing Unit
GPU	Graphics Processing Unit
WEKA	Waikato Environment for Knowledge Analysis
csv	comma separated values
URLs	Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO 8
1.1 OBJETIVOS GERAL E ESPECÍFICOS 9
1.2 JUSTIFICATIVA 10
1.3 ORGANIZAÇÃO DO DOCUMENTO 11
2 REFERENCIAL TEÓRICO 12
2.1 INTELIGÊNCIA ARTIFICIAL E APRENDIZAGEM DE MÁQUINA 12
2.2 PROCESSAMENTO DA LÍNGUA NATURAL 14
2.3 REDES NEURAIS ARTIFICIAIS 15
2.3.1 Ativação dos neurônios 16
2.3.2 Redes perceptron 19
2.3.3 Treinamento de Redes Perceptron 20
2.3.4 Backpropagation 21
2.4 REPRESENTAÇÃO DE PALAVRAS E WORD EMBEDDING 22
2.4.1 Vetores One-Hot 22
2.4.2 Word Embedding 23
2.5 MODELOS DE WORD EMBEDDING 24
2.5.1 CBOW 25
2.5.2 Skip-gram 26
2.6 TEOREMA DE BAYES E O CLASSIFICADOR BAYESIANO INGÊNUO 27
2.6.1 Classificador Bayes Ingênuo 28
3 MATERIAIS E MÉTODOS 30
3.1 MATERIAIS, TECNOLOGIAS E FERRAMENTAS 30
3.2 ETAPAS DE PRÉ-PROCESSAMENTO 32
3.2.1 Tratamento do corpus 32
3.2.2 Cálculo das probabilidades das palavras 33
3.2.3 Obtenção dos Word Embeddings 35
4 EXPERIMENTOS 36
4.1 BAYESIANO INGÊNUO UTILIZANDO O WEKA 36
4.2 VALIDAÇÃO DAS PROBABILIDADES DAS PALAVRAS 37
4.3 CRIAÇÃO E TREINAMENTO DAS RNAS 37
4.4 AVALIAÇÃO DAS RNAS SOBRE AS MENSAGENS DO CORPUS 39
5 RESULTADOS E DISCUSSÕES 41
5.1 EXPERIMENTO COM O BAYESIANO IGNÊNUO UTILIZANDO O WEKA 41
5.2 EXPERIMENTO PARA VALIDAÇÃO DAS PROBABILIDADES CALCULADAS 42
5.3 EXPERIMENTO DE TREINAMENTO DAS RNAS 43
5.4 EXPERIMENTO DE AVALIAÇÃO DAS RNAS SOBRE AS MENSAGENS DO CORPUS 50
6 CONCLUSÕES 54
6.1 TRABALHOS FUTUROS 55
REFERÊNCIAS 56

1 INTRODUÇÃO

Segundo o relatório de mobilidade da Ericsson (2018), 3.5 de 7.8 bilhões das conexões móveis ainda são realizadas de dispositivos simples, que não dispõem das tecnologias "smart", como: grupos de WhatsApp[1] e e-mails, o que deixa de fora mais da metade dos de consumidores de tecnologia móvel do alcance das ferramentas de marketing consideradas mais modernas. O Serviço de Mensagens Curtas (SMS), em contrapartida, é o padrão de comunicação em texto para dispositivos móveis, garantindo que todo esse mercado seja atingido.

Esendex (2018) constatou que na média global, 73% das pessoas abrem todos os SMS que recebem, e, agrupando as pessoas que abrem as mensagens apenas quando conhecem o emissor, esta média alcança 94% das pessoas. Segundo a pesquisa de Zogby Analytics (ZOGBY, 2014), 87% dos *millenials* (pessoas da geração Y, nascidas a partir dos anos 80 até o final dos anos 90) dizem que seus smartphones estão ao seu alcance o tempo todo, e que para 80% deles, visualizá-lo é a primeira coisa a fazer quando acordam.

Consumidores no geral destacaram sentir-se lembrados e gostarem quando a empresa comunicadora mostra se importar com eles, e somente 17% das pessoas que recebem SMS o consideram intrusivo (LOUDHOUSE, 2014).

Com o conhecimento desses dados, muitas empresas podem perceber que o marketing via SMS pode trazer diversos benefícios quanto a divulgação de seus produtos ou serviços, bem como realizar uma aproximação mais pessoal e humanizada com seus clientes.

Existem ferramentas de automatização de envio disponíveis na Internet, que oferecem diversas vantagens ao público, como agendar envios, possibilitar envios em massa para uma lista de contatos e profissionalizar o envio com números que caracterizem o SMS como sendo enviado por uma empresa. Exemplos de ferramentas encontradas são: *trendMens*[2], *Apifonica*[3] e *Zenvios*[4].

Os desenvolvedores dessas ferramentas utilizam-se de Interfaces de Programação para

[1] http://www.whatsapp.com
[2] http://www.trendmens.com.br/
[3] https://www.apifonica.com/products/messaging-api.html
[4] http://www.zenvia.com.br/pt/envio-de-sms/

Aplicações (API) disponibilizadas por plataformas de envio como *Infobip*[5] e *Comtele*[6], que servem como mediadoras entre seus sistemas e as operadoras telefônicas para a entrega dos textos enviados pelo consumidor aos celulares de seus contatos.

Pelo fato dessas ferramentas estarem disponíveis na Internet e oferecerem testes gratuitos, qualquer pessoa que realize um cadastro está disponível para enviar mensagens livremente para um ou mais contatos cadastrados em sua conta, fato esse que permite usuários maliciosos propagar mensagens de spam a diversos celulares de desconhecidos.

Em alguns casos, a plataforma de envio de SMS especifica o uso da API a partir de um contrato firmado com o desenvolvedor da aplicação de envio de mensagens, onde define restrições e punições caso seja detectado o envio de *spam* pela sua aplicação[7]. Porém, as operadoras não oferecem qualquer tipo de filtro que possa barrar essas mensagens, ficando o tratamento do conteúdo a cargo do desenvolvedor do sistema de envio.

Dentre as técnicas utilizadas para filtragem de *spam* (HIDALGO et al., 2006), a classificação de mensagens métodos de aprendizado como os chamados Filtros Bayesianos conseguem alcançar grande precisão. Mas seus resultados são restritos à base de aprendizado, dependendo de parâmetros como tamanho da base e rotulação correta.

O desenvolvimento deste trabalho visa buscar uma maneira de identificar *spam* em mensagens SMS implementando um filtro Bayesiano, e buscar modos de melhorar o resultado obtido aplicando técnicas de aprendizagem de máquina.

1.1 OBJETIVOS GERAL E ESPECÍFICOS

Por meio de aprendizagem de máquina, desenvolver um detector de *spam* para mensagens enviadas por sistemas de envio automático de SMS. O desenvolvimento deste trabalho se dará a partir dos objetivos específicos a seguir:

- Classificar as mensagens SMS de uma base de dados a partir da implementação do algoritmo classificador Bayesiano Ingênuo;
- Verificar se a implementação manual do algoritmo é capaz de se equiparar ao algoritmo classificador presente na ferramenta Weka;

[5]https://www.infobip.com/

[6]https://www.comtele.com.br/

[7]Para este trabalho, a cláusula da multa sobre spam encontra-se em um contrato assinado entre uma empresa consultada, baseada no Brasil, e a plataforma de envio InfoBip, baseada internacionalmente

- Utilizar a representação vetorial das palavras presentes nas mensagens para treinar uma RNA capaz de prever as probabilidades calculadas pelo algoritmo Bayesiano;
- Verificar se a utilização das RNA contribui com a eficiência do classificador ao prever probabilidades de palavras nunca vistas antes.

1.2 JUSTIFICATIVA

O marketing via SMS tem se provado uma grande ferramenta para abrangência da comunicação de empresas com seus clientes. Na pesquisa The SMS Advantage Report desenvolvida pela SAP em conjunto com a Loudhouse Research (LOUDHOUSE, 2014), 64% dos entrevistados acham que as empresas devem usar SMS como forma de aproximação com seus clientes, além de que 70% deles concordam que o uso de marketing via SMS é uma grande forma das empresas ganharem atenção no mercado.

As vantagens do SMS continuam, onde sua taxa de resposta e conversão de clientes é de aproximadamente 45%, muito maior que a taxa de 8% do marketing feito por e-mail (SMSGLOBAL, 2018).

Dados como esses são bons argumentos para empresas entrarem no ramo do marketing via SMS, e pensando nisso, desenvolvedores podem buscar uma fonte de lucros ao fornecer ferramentas que auxiliem as empresas a potencializar o uso desse recurso.

O fato de poder ser penalizado caso uma mensagem seja denunciada como sendo spam é um peso muito grande ao desenvolvedor considerando, que tanto a plataforma de envio quanto a operadora, mesmo tendo detectado o envio, não fornecem nenhuma forma de bloquear esse tipo de mensagem. Já que junto com o crescimento do envio de mensagens legítimas utilizando as mais diversas aplicações, o envio de spam utilizando o mesmo formato também cresce de maneira alarmante (HIDALGO et al., 2006).

Formas de bloqueio como colocar palavras chave em listas de proibição (Blacklist e Whitelist) são uma maneira rápida de remediar o problema, porém não são eficientes, pois uma lista deve ser mantida manualmente, e utilizar esta técnica pode bloquear mensagens que não são spam, pois palavras em blacklist podem estar contidas nela, sem que o contexto completo da mensagem seja um spam (ZHANG; WANG, 2009).

Dentre as técnicas que melhor apresentam precisão na detecção de spam, está a Aprendizagem de Máquina. Por possuir forte base teórica em matemática e grande capacidade

compreensiva com dados, as técnicas de filtragem com os algoritmos Bayesianos são grande foco no aprendizado de máquina para esse tipo de filtro (ZHANG; WANG, 2009; ALMEIDA et al., 2013).

Este trabalho se baseia na utilização das técnicas de Aprendizagem de Máquina, para criar uma forma de amplificar a identificação de *spam* em mensagens SMS, permitindo assim, que o bloqueio de seu envio seja realizado de maneira mais eficiente.

1.3 ORGANIZAÇÃO DO DOCUMENTO

Este documento está organizado da seguinte forma: o Capítulo 2 introduz as áreas de conhecimento abordadas no desenvolvimento deste trabalho: Inteligência Artificial, Processamento de Língua Natural e Redes Neurais Artificiais. Em seguida, no Capítulo 3 são apresentados os recursos, ferramentas e a metodologia utilizada para a construção dos experimentos. No Capítulo 4 estão definidos os experimentos realizados neste trabalho. No Capítulo 5 são discutidos os resultados obtidos através da realização dos experimentos. Por fim, no Capítulo 6 apresenta-se as considerações e conclusões deste trabalho.

2 REFERENCIAL TEÓRICO

Neste capítulo são apresentados os campos de estudo presentes na Inteligência Artificial que serão utilizados como a base para o desenvolvimento deste trabalho.

2.1 INTELIGÊNCIA ARTIFICIAL E APRENDIZAGEM DE MÁQUINA

A Inteligência Artificial (IA) é a área da Ciência da Computação onde seus estudos voltam-se para o desenvolvimento de sistemas computacionais inteligentes, ou seja, sistemas que apresentam características associadas à inteligência no comportamento humano, tais como compreensão da linguagem, aprendizado, raciocínio e resolução de problemas (FERNANDES, 2003).

Dentre as competências da IA (MONARD; BARANAUSKAS, 2003), a Aprendizagem de Máquina (AM) está ligada ao desenvolvimento de técnicas computacionais sobre o aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento de maneira automática. Um sistema de aprendizado é um programa de computador capaz de tomar decisões baseado em experiências acumuladas na solução de problemas anteriores.

Aprendizagem de Máquina é uma sub-área muito importante de pesquisa em IA, pois a capacidade de aprender é essencial para um comportamento inteligente (BATISTA, 2003).

Para Mitchell (1997), AM está ligada a questão de como construir computadores que melhoram seu desempenho em alguma tarefa através de experiências. Ainda, segundo ele, algoritmos de AM tem provado excelentes resultados práticos em diversos domínios, como (a) problemas na mineração de dados onde grandes conjuntos de dados apresentam padrões implícitos que possam ser úteis; (b) domínios ainda pouco compreendidos onde humanos podem não possuir o conhecimento necessário para construir algoritmos (por exemplo, reconhecimento facial em imagens); (c) domínios onde o programa deve se adaptar dinamicamente à condições variáveis (como adaptar os interesses de leitura de uma pessoa);

e (d) domínios onde o custo para adquirir ou codificar conhecimento manualmente é muito custoso, como é o caso do Processamento da Língua Natural (PLN).

A hierarquia do aprendizado (Figura 1) presente na AM descreve as maneiras possíveis a um sistema de aprendizado conseguir organizar sua forma de reunir conhecimento e poder de tomada de decisões.

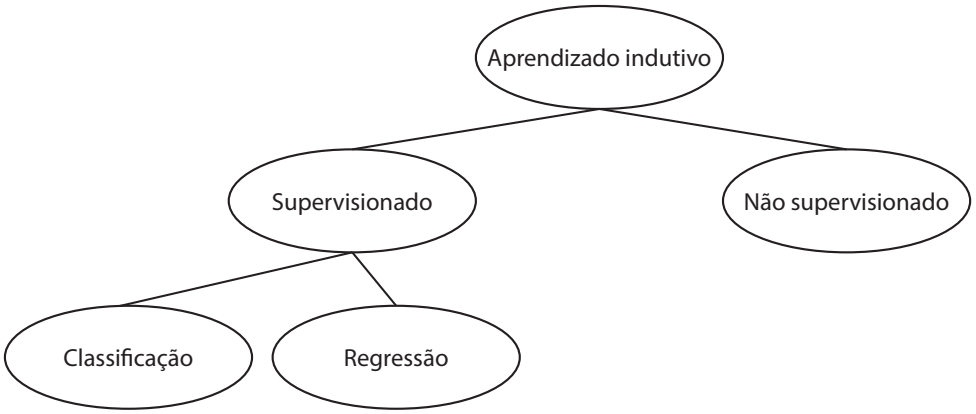


Figura 1 – A Hierarquia do aprendizado.
Fonte: Adaptado de Batista (2003)

A inferência indutiva é um dos principais meios de criar novos conhecimentos e prever futuros (BATISTA, 2003). A indução é uma forma de inferência lógica que permite tirar conclusões genéricas sobre um conjunto particular de exemplos. Caracteriza-se como o raciocínio que se origina em um conceito específico e o generaliza, ou seja, a partir do particular se chega ao universal (MONARD; BARANAUSKAS, 2003).

Divide-se o aprendizado indutivo entre *supervisionado* e *não-supervisionado*. No aprendizado supervisionado, fornece-se o algoritmo de aprendizado, o *indutor* e um conjunto de exemplos de treinamento rotulado conforme a classe a qual o exemplo pertence. O número de exemplos, entretanto, não é suficiente para o sistema caracterizar completamente a similaridade, assim, o sistema é apenas capaz de induzir uma hipótese que aproxima novos exemplos à classe a qual eles poderiam pertencer (BATISTA, 2003). O termo supervisionado denota o fato que os valores de saída para exemplos de treinamento são conhecidos (LACERDA; BRAGA, 2004).

Segundo Lacerda e Braga (2004), dentro das tarefas competentes ao aprendizado supervisionado, estão:

- Classificação: a partir de um conjunto de treinamento contendo instâncias de classes com atributos conhecidos, reconhecer as características e padrões que possam ser utilizados como fronteiras de decisão para reconhecer a classe de novas instâncias;
- Regressão: estimação de funções contínuas que possam estabelecer relacionamentos entre uma variável dependente e uma ou mais variáveis independentes.

No aprendizado não supervisionado, fornece-se ao sistema um conjunto de exemplos, consistindo apenas de características dos elementos, sem qualquer informação sobre a classe a qual cada exemplo pertence. O objetivo do aprendizado não supervisionado é construir um modelo que procura por padrões, agrupando exemplos que possuem características parecidas (BATISTA, 2003).

2.2 PROCESSAMENTO DA LÍNGUA NATURAL

PLN é o nome dado à área de pesquisa dedicada a investigar, desenvolver modelos, técnicas e sistemas computacionais que possuem a língua natural como objeto de estudos, bem como explorar como computadores podem ser usados para entender e manipular a língua natural em forma de texto e fala (CHOWDHURY, 2003).

Quando fala-se em processamento de língua e fala, o termo está ligado a qualquer técnica computacional que processa a linguagem humana escrita e falada, ou seja, engloba desde as mais simples aplicações como contadores de palavras em documento, até complexas aplicações como tradução de língua falada em tempo real (JURAFSKY; MARTIN, 2009).

O que difere estas aplicações é o seu uso do "conhecimento da linguagem". Por exemplo, um sistema de contagem de palavras, que é usado para numerar bytes, palavras e linhas num texto. Quando é usado apenas para contagem de bytes, é um simples processador de informação, no entanto, quando usado para contar as palavras em um arquivo, o sistema deve ter conhecimento sobre "o que significa ser uma palavra", tornando-se um sistema processador de linguagem (JURAFSKY; MARTIN, 2009).

O conhecimento necessário para entender uma linguagem (JURAFSKY; MARTIN, 2009), pode ser classificado em seis categorias:

- Fonética e Fonologia: o estudo e interpretação dos sons da língua;
- Morfologia: o estudo dos componentes que formam as palavras;
- Sintaxe: o estudo das relações estruturais entre palavras;
- Semântica: o estudo do sentido;
- Pragmático: o estudo de como a língua é utilizada para alcançar objetivos;
- Discurso: o estudo da produção e estruturação de textos.

Quanto ao PLN, para aplicar tais conhecimentos destaca-se as tarefas de pré-processamento de textos, classificar (etiquetar) e mapear as representações da língua (NUNES,

2008).

2.3 REDES NEURAIS ARTIFICIAIS

Em 1943, Warren McCulloch e Walter Pitts apresentaram o primeiro modelo de um neurônio artificial, a partir de circuitos digitais e valores binários como forma de ativação de entrada e como respostas de saída. Com o passar do tempo, o campo de pesquisa em redes neurais tem crescido e recebido cada vez mais atenção, assim novas e mais complexas propostas de modelos tem sido desenvolvidas (ROJAS, 1996).

Como forma de funcionamento, as Redes Neurais Artificiais (RNA) utilizam-se de uma maciça interligação entre neurônios artificiais, de maneira que possuam a propensão natural de armazenar conhecimento e experiência para uso, para (HAYKIN, 2001), uma RNA assemelha-se ao cérebro humano em dois aspectos:

- O conhecimento é adquirido num processo de aprendizagem;
- Sinapses entre os neurônios são a forma de armazenar e utilizar o conhecimento.

Em 1958, Franklin Rosenblatt propôs o modelo *Perceptron*, baseado no modelo de McCulloch e Pitts. O grande diferencial desse modelo é aplicar a ponderação dos pesos nas entradas de informações para cada neurônio (ROJAS, 1996).

Para alcançar os objetivos designados à rede, os algoritmos de aprendizado são responsáveis por modificar os pesos das sinapses entre os neurônios para adaptar e melhorar o processo do aprendizado (HAYKIN, 2001).

A Figura 2 representa o modelo de funcionamento de um neurônio artificial, o principal componente de uma RNA, onde $(x_1, x_2, ..., x_n)$ são os valores dos sinais numéricos de entrada, os pesos que ponderam esses valores são $(w_1, w_2, ..., w_n)$, a ponderação determina qual a contribuição que cada entrada possui no resultado de saída do neurônio, o círculo representa o neurônio em si, nele os valores ponderados são somados e acrescentados ao bias θ, que está relacionado ao limiar de ativação do neurônio e afeta seu disparo de informações, dando como retorno o valor u. Se o valor de u superar o limiar de ativação, ele é aplicado à função de ativação $g(u)$. O resultado da função de ativação é o valor de saída y do neurônio artificial (FINNOCCHIO, 2014).

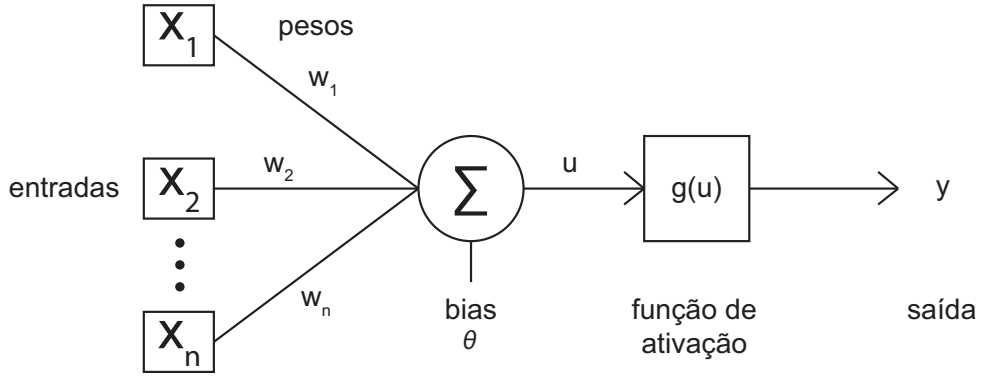


Figura 2 – O *Perceptron*.
Fonte: Adaptado de Finnocchio (2014)

2.3.1 Ativação dos neurônios

Em uma RNA, após a entrada dos valores é iniciado o processo de ativação do neurônio. Como representado na Equação 1, soma-se o limiar de ativação do neurônio com o somatório de cada valor numérico de entrada, ponderado conforme o peso presente na sua ligação com o neurônio. Essa combinação, representado como u, aplicada à função de ativação presente na rede, determina se a informação recebida e processada pelo neurônio será ou não passada adiante (HAYKIN, 2009).

$$u = \theta + \sum_{i=1}^{n} x_i w_i \tag{1}$$

Assim como o funcionamento de entrada e processamento de informações do neurônio artificial estão inspirados na biologia de um neurônio real, a propagação de informações também segue o mesmo princípio, a função de ativação de um neurônio artificial, baseada na soma das entradas, determina se a informação processada será enviada adiante, tal como um neurônio real pode ou não realizar uma sinapse, baseado na carga elétrica recebida.

Para a aplicação artificial, faz-se necessária a utilização de funções matemáticas que possam efetivamente simular uma ativação. Em um perceptron, tem-se como exemplo de ativação uma adaptação da função sinal, chamada de função passo, representada na Equação 2 e visualizada na Figura 3 (MITCHELL, 1997).

$$y = g(u) = \begin{cases} 1, & \text{se } u \geq 0 \\ 0, & \text{caso contrário} \end{cases} \tag{2}$$

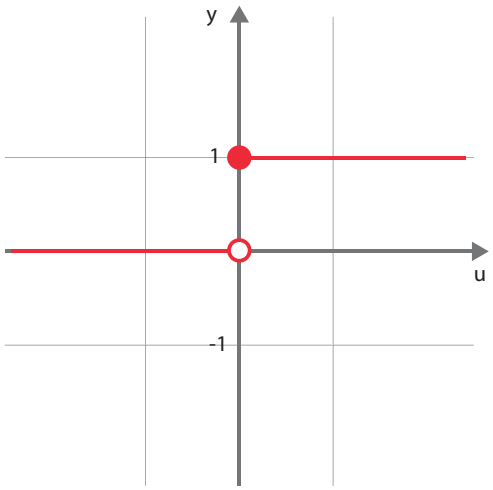


Figura 3 – Visualização gráfica da função passo.

Fonte: Autoria própria

O funcionamento da função passo se dá de maneira simples: o seu parâmetro de funcionamento é a combinação u realizada pelo neurônio, se o valor de u for maior ou igual a zero, a função passo terá um valor unitário na sua saída y, isso significa a propagação do sinal pelo neurônio, caso contrário, a função terá saída y nula, significando a inibição do sinal por parte do neurônio.

Outras funções mais sofisticadas são utilizadas na ativação de um neurônio, buscando unir a similaridade biológica com a melhoria de modelos para treinamento e aprendizado das RNA. Pode-se destacar entre estas funções a função Sigmoide Logística e a função tangente hiperbólica (HAYKIN, 2009). Essas funções são chamadas de sigmoides, por apresentarem formas semelhantes à letra "s" em sua visualização gráfica. Das principais características que as diferem da função passo estão o fato de serem contínuas, não lineares e diferenciáveis, tais atributos são de grande interesse na aplicação de alguns algoritmos de aprendizado, como poderá ser visto na subseção 2.3.

A função Sigmoide Logística é expressada tal como na Equação 3 e visualizada na Figura 4, onde a é um valor escalar arbitrário positivo e $u_j(n)$ é o campo local de indução do neurônio j (HAYKIN, 2009).

$$y = sigmoide(u) = \frac{1}{1+e^{-au_j(n)}}, \qquad a > 0 \tag{3}$$

A função tangente hiperbólica definida como na Equação 4 e visualizada na Figura 5, difere-se graficamente da Sigmoide logística onde seu intervalo de saídas y é assintótico aos valores unitários positivo e negativo (HAYKIN, 2009).

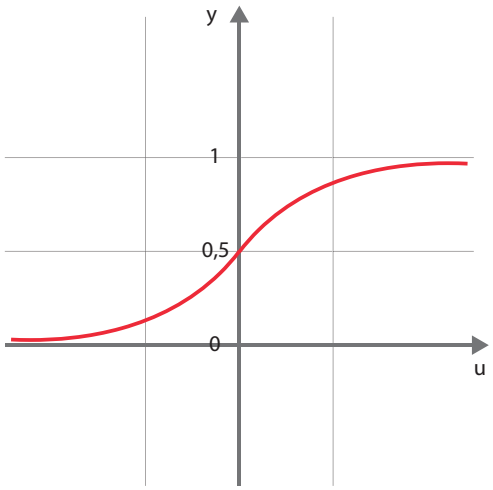


Figura 4 – Visualização gráfica da função Sigmoide Logística.

Fonte: Autoria própria

$$y = tanh(u) = \frac{e^{u} - e^{u}}{e^{u} + e^{u}} \tag{4}$$

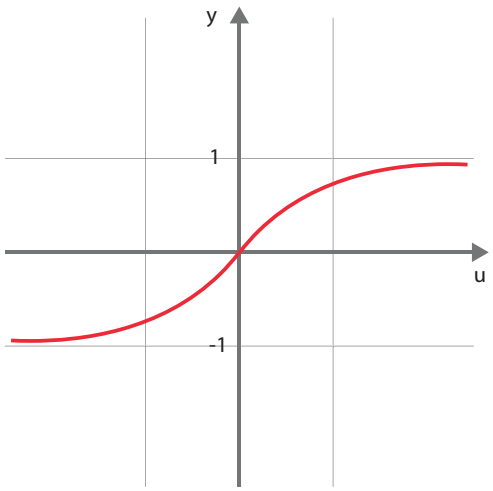


Figura 5 – Visualização gráfica da função tangente hiperbólica.

Fonte: Autoria própria

2.3.2 Redes perceptron

As RNA do tipo perceptron podem existir como uma camada única (neurônio Perceptron) e em múltiplas camadas (Multilayer Perceptron). Quando a rede é de camada única, possui apenas a entrada e os neurônios são a própria saída (Figura 2), sua aplicação só é possível em problemas onde os objetos distribuídos no espaço de instâncias são linearmente separáveis, como mostrado na Figura 6.

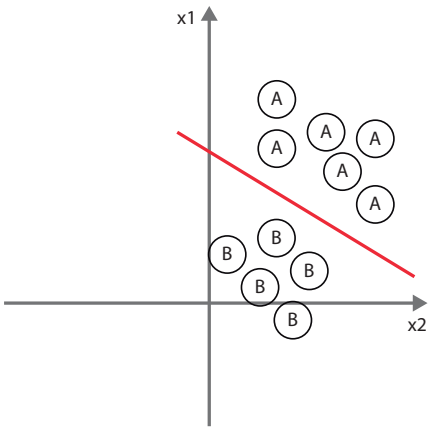


Figura 6 – Problema linearmente separável.

Fonte: Autoria própria

A distinção para a rede multicamada é a presença de uma ou mais camadas escondidas de neurônios, sendo que a saída dos neurônios internos são os valores de entrada para os neurônios da camada seguinte (Figura 7). Assim, o aprendizado passa a depender também dos resultados gerados por outros neurônios, permitindo a resolução de problemas onde as instâncias não separáveis linearmente no espaço, como mostrado na Figura 8 (FINNOCCHIO, 2014).

Consistindo de conjuntos sensoriais (camada de entrada), camadas ocultas (no caso do Multilayer Perceptron) e camadas de saída, seu funcionamento básico de treinamento, conhecido como *feedforward* é dado pela propagação do sinal em sentido único para frente entre os componentes da rede (HAYKIN, 2001).

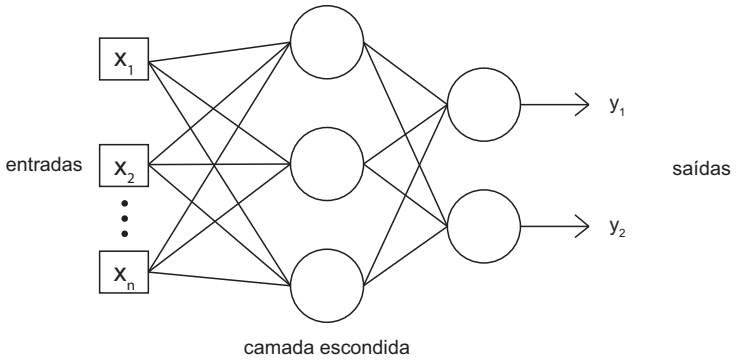


Figura 7 – Rede feedforward multicamadas (com uma camada escondida).

Fonte: Adaptado de Finnocchio (2014)

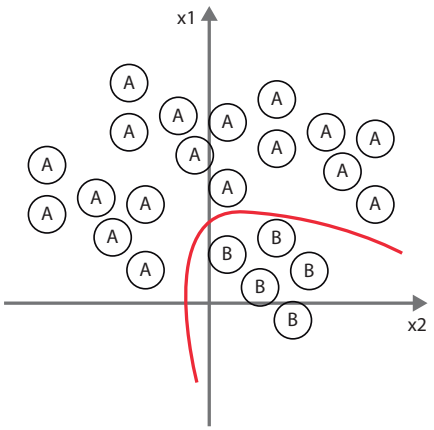


Figura 8 – Problema linearmente separável.

Fonte: Autoria própria

2.3.3 Treinamento de Redes Perceptron

O problema do treinamento de Redes Perceptron é o de conseguir realizar os ajustes necessários nos pesos de ponderação da rede, para que as saídas obtidas sejam iguais às saídas desejadas da rede (MITCHELL, 1997).

Uma das maneiras de realizar este treinamento é escolher aleatoriamente pesos para as conexões, e aplicar o perceptron para cada exemplo de treino, modificando os pesos após cada época de treinamento. Uma época consiste no treinamento de todos os elementos do conjunto de testes na rede perceptron. Os ajustes são realizados até que o perceptron consiga classificar corretamente cada exemplo, ou seja, cada saída obtida é igual à saída desejada em cada treino. Os pesos w_i são modificados em relação à entrada x_i de acordo com regra de treinamento de

Perceptron (MITCHELL, 1997):

$$w_i = w_i + \eta(d - y)x_i \tag{5}$$

Onde d é o valor desejado de saída da rede e y é o valor de saída obtido na iteração anterior, sua diferença caracteriza o erro da rede.

O termo η refere-se à taxa de aprendizado proposta para a RNA. A taxa de aprendizado pode ser considerada uma guia para o ajuste dos pesos durante a classificação das instâncias do problema. Quanto menor a taxa de aprendizado, mais suave será o ajuste de separação do problema no espaço de instâncias, porém o custo computacional pode ser muito alto. Em contrapartida, se a taxa definida for muito alta, buscando agilidade no processo do aprendizado, a mudança entre os ajustes dos pesos pode oscilar exageradamente, tornando o aprendizado da rede muito instável (HAYKIN, 2009).

2.3.4 Backpropagation

Em redes Multilayer Perceptron, pode-se ocorrer duas fases de aprendizado, a primeira fase, conhecida como "para frente" (*feedforward*) como exemplificado na Figura 7, os neurônios das camadas internas recebem suas entradas ponderadas e produzem suas saídas, suas saídas então servem como entrada pra os neurônios da camada seguinte, até que a camada de saída produza seu valor. Ao final deste processo, calcula-se o erro produzido pela rede pela diferença entre os valores de saída final e os valores desejados que a rede produza (MITCHELL, 1997).

Na segunda fase, chamada de "para trás" *backward*, o erro calculado serve como base para a alteração nos pesos de ponderação das camadas, que vão desde a saída até a primeira camada interna da rede. O algoritmo *Backpropagation*, aplicando as duas fazes, faz com que o erro de uma camada intermediária seja estimado com os erros da camada seguinte (MITCHELL, 1997).

Utilizando como exemplo uma RNA onde a ativação é dada pela função Sigmoide Logística, o funcionamento da fase *backward* algoritmo pode ser exemplificado da seguinte forma:

Para os neurônios da camada de saída, o ajuste dos pesos é dado pela Equação 6.

$$\Delta w_{ji} = \eta(d_j - y_j)y_j(1 - y_j)x_{ji} \tag{6}$$

Onde w_{ji} denota o peso relacionado à ligação i ao neurônio j, x_{ji} indica a $i$ésima

entrada no neurônio j, η é a taxa de aprendizado, e $(d_j - y_j)y_j(1-y_j)x_{ji}$ é obtido diferenciando da função de ativação (MITCHELL, 1997).

Para os neurônios das camadas intermediárias, o ajuste dos pesos é dado pela Equação 7. Leva-se em conta a influência do peso nos erros de saídas posteriores na rede, o fluxo posterior ao peso w_j será denotado por F.

$$\delta w_j = y_j(1-y_j)\sum_{k \in F} \delta_k w_{kj} \tag{7}$$

A diferenciação da função de ativação é realizada para o tratamento do custo do algoritmo, para isso faz-se necessário utilizar uma função diferenciável, por isso da importância da aplicação das funções Sigmoide como ativação numa RNA (HAYKIN, 2009).

2.4 REPRESENTAÇÃO DE PALAVRAS E WORD EMBEDDING

Como em uma RNA os valores de entrada são numéricos, e muitas tarefas dentro do PLN consistem no tratamento e análise de palavras, é necessário encontrar uma maneira de representar as palavras como dados numéricos para possibilitar seu manuseio em uma RNA, a representação vetorial é a mais comumente utilizada.

2.4.1 Vetores One-Hot

Uma das maneiras mais simples de representação vetorial das palavras é discretizá-las como um símbolo atômico em um vetor, onde cada posição ou atributo é uma palavra diferente (GOLDBERG, 2015). Cada atributo é completamente independente do outro, sendo assim, “cachorro” é tão diferente de “democracia” quanto é diferente de “gato”.

A Figura 9 exemplifica a representação vetorial das palavras dentro do modelo One-Hot, onde a posição contendo o valor 1 (um) indica o índice *i* da palavra ordenada segundo o vocabulário, e as demais posições são preenchidas com o valor 0 (zero) (CHAUBARD et al., 2016).

Por serem representados de maneira independente, os vetores de cada palavra possuem

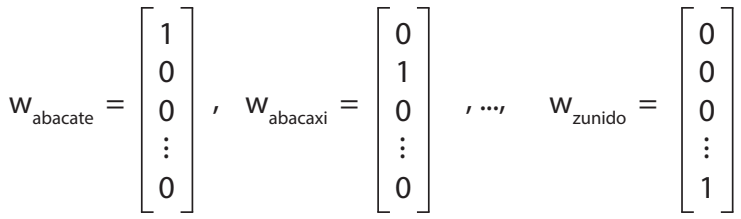


Figura 9 – Representação One-Hot.
Fonte: Adaptado de Chaubard et al. (2016)

dimensão de tamanho equivalente ao nosso vocabulário, sendo computacionalmente muito custosos de se trabalhar. RNAs tem sua eficiência comprometida quando aplicadas à vetores esparsos e de dimensões muito elevadas devido a chamada *maldicação da dimensionalidade* (GOLDBERG, 2015). Além do mais, tal representação não consegue capturar relações de similaridade entre as palavras.

$$(W_{democracia})^T.W_{cao} = (W_{democracia})^T.W_{gato} = (W_{cao})^T.W_{gato} = 0 \tag{8}$$

Na Equação 8, considerando que "cão" e "gato" compartilham atributos em comum, tais qual: serem animais, animais de estimação, quadrúpedes e mamíferos, pela representação One-Hot sua dissimilaridade é tão igual quanto compará-los com "democracia", então o mais adequado a se buscar, seria encontrar uma forma de representação menor e menos custosa, e que além disso possa apresentar sub-espaços onde possa-se encontrar algum tipo de relação e semelhança entre palavras (CHAUBARD et al., 2016).

2.4.2 Word Embedding

Para que o processamento de linguagem natural tenha todo seu potencial explorado por uma RNA, deseja-se ter à disposição modelos de palavras que possam fornecer um grande apoio ao aprendizado e aplicação do PLN, provendo uma grande variedade de informações que possam ser úteis nas mais diversas análises.

Segundo Dhillon et al. (2015), os modelos de dados geralmente analisados por sistemas de PLN são rotulados, já dando um grande norte sobre como o sistema deve se comportar na aplicação desses modelos. Porém na maior parte dos casos há apenas uma quantidade limitada de rótulos disponíveis para esta tarefa, principalmente quando se trata de aprendizados

em línguas que não sejam a língua inglesa, dificultando a performance do aprendizado. Em contrapartida, *embeddings* de palavras, que são aprendidos a partir de uma massiva quantidade de dados não rotulados provêm conjuntos de dados altamente discriminativos para aprendizado, melhorando o desempenho de sistemas de PLN.

Word Emeddings mapeiam cada palavra em um vetor k dimensional de valores contidos no conjunto dos números reais $\mathbb{R}$. Esse mapeamento se dá de modo não supervisionada, analisando a co-ocorrência de palavras em conjuntos não rotulados. Normalmente a dimensão k dos vetores gerados para as palavras está no intervalo de $[50, 300]$, muito menor do que a representação One-Hot, que possui a dimensão de todo vocabulário de uma língua (DHILLON et al., 2015).

A representação de uma palavra aprendida, chamada *embedding*, é mapeada em um vetor como $vec(cachorro) = (0.6, 0.78, -0.12, ..., 3.1)$. Assim as representações computadas e aprendidas por uma RNA consegue encontrar padrões e regularidades linguísticas. Esses padrões podem ser representados através de operações matemáticas nos vetores. A Equação 9 apresenta um dos padrões que podem ser aprendidos em uma rede (MIKOLOV et al., 2013).

$$vec(Madrid) - vec(Espanha) + vec(França) = vec(Paris) \tag{9}$$

Tal padrão é aprendido, pois o resultado da equação $vec(Madrid) - vec(Espanha) + vec(França)$, que é um novo vetor, possui valores que se aproximam de $vec(Paris)$ mais do que qualquer outro vetor de palavras (MIKOLOV et al., 2013).

2.5 MODELOS DE WORD EMBEDDING

Nesta seção serão apresentados os modelos neurais de linguagem CBOW e Skip-Gram, que tentam predizer uma ou mais palavras por meio de uma RNA, dado uma palavra ou um conjunto de palavras.

2.5.1 CBOW

O modelo CBOW (Continuous Bag of Words) tem como objetivo encontrar uma palavra chave a partir de um contexto onde ela está inserida. O contexto é fornecido por palavras de entrada no modelo tendo como saída a palavra chave baseada na previsão ou probabilidade dela aparecer acompanhada no contexto das palavras inseridas. Por exemplo, tendo a entrada "O", "gato", "sobre" e "poça" como contexto, a previsão de saída mais provável a se obter deveria ser "pulou" (CHAUBARD et al., 2016).

Como mostrado na Figura 10, as informações de entrada e saída no modelo CBOW são palavras representadas por vetores One-Hot, sendo as entradas denotadas por $p(t \pm i)$, onde i é definido como a janela de contexto para a palavra desejada e a saída é representada por $p(t)$.

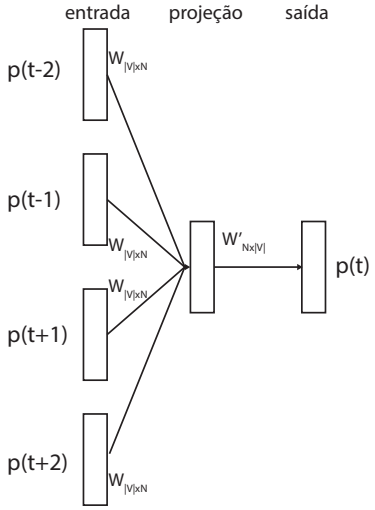


Figura 10 – Modelo CBOW.
Fonte: Adaptado de Chaubard et al. (2016)

As entradas conectam-se à camada oculta através da matriz $W_{|V|xN}$, onde o número N de colunas é um valor arbitrário de características e o número de linhas $|V|$ são os vetores das palavras do vocabulário.

Os vetores de entrada são multiplicados pela matriz W, e em seguida tira-se a média do resultado, tendo como representação um vetor único de *embedding* das palavras. O vetor construído na camada oculta é então multiplicado pela matriz $W'_{Nx|V|}$, retornando um vetor de saída $p(t)$, contendo a pontuação para as palavras. Utilizando a função Softmax, a pontuação é transformada em probabilidades, que por sua vez tentam igualar-se à probabilidade real das palavras. A probabilidade encontrada pela função Softmax resulta em um vetor One-Hot que

representa exatamente a palavra que mais se insere no contexto apresentado pelas entradas (CHAUBARD et al., 2016).

2.5.2 Skip-gram

Modelo proposto por Mikolov et al. (2013), cujo objetivo é encontrar palavras que possam ser úteis em predizer palavras que estejam em seus arredores, ou seja, a partir de uma palavra chave, encontrar um contexto onde ela possa ser inserida, trazendo palavras e significados que possam ser relacionadas em uma frase ou texto. Por exemplo, em uma rede treinada, se a palavra inserida for “Soviética”, a probabilidade das saídas serem “Rússia” e “União” é muito maior do que palavras não relacionadas, como “melancia” ou “canguru”.

O funcionamento do Skip-gram pode ser entendido como um modelo espelhado do CBOW, enquanto ambos são modelos preditivos de palavras, o Skip-gram por sua vez busca trazer um contexto que possa estar relacionado à palavra de entrada (JURAFSKY; MARTIN, 2009).

Conforme a Figura 11, a informação de entrada do modelo é a representação One-Hot da palavra $p(t)$ a ser analisada, e as saídas $p(t \pm i)$ são as representações One-Hot das palavras de contexto encontradas pelo modelo.

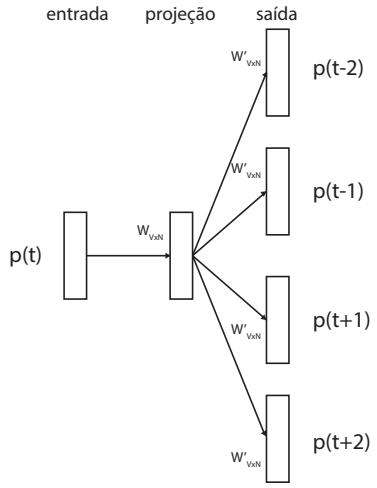


Figura 11 – Skip-gram.

Fonte: Adaptado de Chaubard et al. (2016)

A conexão da entrada com a camada oculta se dá pela matriz $W_{|V|xN}$ onde N é o número

de dimensão e $|V|$ é o tamanho do vocabulário. Essa conexão se dá pela multiplicação do vetor de entrada com a matriz W, gerando um vetor com N posições, que é recebido pela camada oculta.

A matriz $W'_{Nx|V|}$ conecta a camada escondida com a saída, multiplicando o vetor para cada palavra de contexto, para o resultado gerado em pontuação de cada palavra, é calculada sua probabilidade de estar presente no contexto da palavra analisada (JURAFSKY; MARTIN, 2009). Além a função Softmax também ser aplicada no Skip-gram, uma das maneiras mais ágeis e simples de se calcular a probabilidade, é assumir que cada palavra de contexto é independente, podendo-se realizar o cálculo com o classificador Bayes Ingênuo, apresentado na Seção 2.6.

2.6 TEOREMA DE BAYES E O CLASSIFICADOR BAYESIANO INGÊNUO

Em AM, segundo Mitchell (1997), o principal interesse está em determinar a melhor hipótese que generaliza o conhecimento de uma base de dados, isto é, a que melhor prediz dados novos. A hipótese pode ser interpretada geometricamente como uma separação no espaço de instâncias, nas quais as instâncias são distribuídas. As hipóteses, por sua vez, são geometricamente distribuídas dentro de um espaço de hipóteses H. Assim, deseja-se obter-se a hipótese mais provável de estar correta, sendo o conjunto de treinamento D qualquer informação unida a conhecimento prévio sobre as probabilidades das hipóteses presentes em H.

O Teorema de Bayes fornece uma maneira de calcular essa probabilidade. Mais precisamente, uma maneira de calcular a probabilidade de uma hipótese baseada em probabilidades anteriores, probabilidades analisando dados fornecidos sobre a hipótese em questão, e sobre as informações em si (JOYCE, 2016).

Usando $P(h)$ para denotar a probabilidade do evento h ocorrer antes de qualquer treinamento. $P(h)$ pode ser chamado de *probabilidade a priori* de h, referindo-se a conhecimentos anteriores possuídos sobre h poder ser a hipótese correta. Se possuímos qualquer conhecimento anterior, então podemos simplesmente assinalar a mesma probabilidade anterior à cada hipótese candidata. Da mesma forma $P(D)$ refere-se à probabilidade anterior do conjunto de informações para treinamento D ser observado. Seguindo, $P(D|h)$ indica a probabilidade de observar informações contidas em D dado um contexto contido em h (MITCHELL, 1997).

Define-se a probabilidade condicional de um evento A tendo ocorrido um evento B (sendo a probabilidade de B diferente de zero) como mostrado pela Equação 10 (LACERDA;

BRAGA, 2004).

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{10}$$

Onde $P(A \cap B)$ é a probabilidade de ocorrência simultânea dos eventos A e B, ou seja, a probabilidade conjunta de A e B.

Para $P(A) \neq 0$, pode-se escrever também:

$$P(B|A) = \frac{P(B \cap A)}{P(A)} \tag{11}$$

Combinando as Equações 10 e 11, tem-se a forma principal do Teorema de Bayes, mostrado na Equação 12.

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{12}$$

No contexto de AM (MITCHELL, 1997), o interesse está voltado para a probabilidade $P(h|D)$ que h possui dado o conjunto de informações D. $P(h|D)$ é chamada de probabilidade posterior (*posteriori*) de h, pois esta informa a confiança que h possui após as informações em D terem sido analisadas. A probabilidade *posteriori* $P(h|D)$ reflete a influência do conjunto de informações D, em contraste com a probabilidade *a priori* $P(h)$, que é independente de D.

O Teorema de Bayes é o pilar dos métodos de aprendizado Bayesiano pois fornece uma maneira de calcular a probabilidade *posteriori* $P(h|D)$, partindo da probabilidade *a priori* $P(h)$, unido com $P(D)$ e $P(D|h)$. Assim, substituindo na Equação 12 conforme a notação de hipóteses, tem-se:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{13}$$

2.6.1 Classificador Bayes Ingênuo

Um dos métodos mais práticos de aprendizado Bayesiano, é o classificador Bayes Ingênuo (*Naïve Bayes Classifier*). Este método se mostra bastante eficiente quando utilizado na classificação de textos. (MITCHELL, 1997).

Quando desconhece-se as relações de dependência entre os dados de entrada utilizados por um classificador, supõe-se que todos os dados são independentes dado sua categoria

(LACERDA; BRAGA, 2004). Assim serão dados como:

$$D = (D_1, ..., D_n)^T \tag{14}$$

Então:

$$P(D|h) = P(h) \prod_{i=1}^{n} P(h|D_i) \tag{15}$$

Aplicando a fórmula da Equação 15 na Equação de Bayes (Equação 13), temos a chamada regra *Naïve Bayes*. Por mais que essa seja uma posição ingênua a se tomar quanto a uma decisão baseada em probabilidades, o classificador ingênuo tem mostrado sucesso notável comparado a outros métodos e tem provado grande eficiência em áreas como classificação textual, diagnósticos médicos e sistemas de gerenciamento de desempenho (RISH, 2001).

3 MATERIAIS E MÉTODOS

Neste capítulo são descritos os materiais utilizados, bem como a definição e descrição dos métodos e experimentos empregados na realização deste trabalho.

3.1 MATERIAIS, TECNOLOGIAS E FERRAMENTAS

O conjunto de exemplos para treinamento utilizado neste trabalho é o *SMS Spam Collection v.1* em língua inglesa composto pela reunião de 5.574 mensagens SMS extraídas de diversas fontes e rotuladas como sendo *spam* ou *ham* (mensagens legítimas) afim de proporcionar os exemplos necessários para o desenvolvimento do aprendizado classificador (ALMEIDA et al., 2013).

A ferramenta utilizada para o desenvolvimento do código para os métodos de classificação foi o editor de textos Visual Studio Code[1]. Escolhido pela familiaridade e também por ser um editor de textos simples, leve e que fornece de maneira fácil a personalização e organização tanto do ambiente quanto do projeto, agilizando o processo de desenvolvimento.

A linguagem de programação escolhida para desenvolvimento dos códigos foi a linguagem Python[2]. Sua escolha foi dada pela popularidade da linguagem no meio da IA, e por fornecer bibliotecas para desenvolvimento de trabalhos com PLN, como NLTK e fastText. A linguagem também pode ser executada nos principais sistemas operacionais, permitindo maior portabilidade e possibilidade de replicação dos resultados do trabalho.

A biblioteca pandas[3], disponível para a linguagem Python, fornece estruturas de dados que facilitam a análise e manipulação de dados.

A biblioteca Natural Language Toolkit[4], ferramenta para Python, fornece fácil

[1] https://code.visualstudio.com
[2] https://www.python.org/
[3] https://pandas.pydata.org
[4] http://www.nltk.org/

manipulação de funções para trabalho com língua natural, como Tokenização, classificação, remoção de *stop words* (palavras que não agregam valor de significado para um contexto, no caso da classificação das mensagem SMS, exemplos são as preposições e os artigos), análise de sentimentos e rotulação.

A biblioteca fastText[5], desenvolvida visando a manipulação de textos, extremamente eficiente quando aplicada em coleções de larga escala. Com um dos grandes focos de sua aplicação voltados para o PLN, e implementação em linguagem Python, fornece métodos para a utilização dos modelos de Word Embedding, como Skip-Gram e CBOW.

O TensorFlow[6] é uma ferramenta utilizada para escrever algoritmos de Aprendizado de Máquina baseado em grafos de fluxos, onde os nós do grafo são operações matemáticas realizadas sobre as arestas, cada aresta consiste de um *tensor* (matriz multidimensional de dados). Os cálculos podem ser realizados sobre a CPU (Central Processing Unit) ou GPU (Graphics Processing Unit) da máquina.

Python foi a primeira linguagem suportada por TensorFlow, possuindo mais funcionalidades quando comparada com outras linguagens. Tanto a escolha do TensorFlow quando do Python pesaram uma sobre a outra dado este fato.

A API Keras[7], ferramenta de alto nível para criação de modelos de RNA, utiliza TensorFlow (ou outras ferramentas, como CNTK ou Theano) para facilitar e agilizar a escrita e experimentação com redes neurais artificiais, permitindo execuções a partir de CPU e GPU, em redes desde as mais simples até redes convolucionais e de Aprendizado Profundo.

A ferramenta WEKA[8] (Waikato Environment for Knowledge Analysis) foi escolhida por conter algoritmos de pré-processamento de dados, classificação, regressão e regras de associação, que podem ser úteis para realizar análises do *corpus* em métodos já existentes e compará-los aos resultados obtidos no modelo desenvolvido neste trabalho.

O *corpus* de SMS também é disponibilizado em um arquivo que de extensão compatível para leitura e utilização na ferramenta WEKA, podendo ser usado para testes de classificação com diversos algoritmos presentes na ferramenta, permitindo a obtenção de resultados que possam ser utilizados posteriormente como comparativos para com o desempenho da solução desenvolvida neste trabalho.

O ambiente para desenvolvimento deste trabalho foi um notebook Apple MacBook Pro com sistema operacional macOS na versão 10.13.4 High Sierra, processador Intel Core i7 com frequência de 2.6 GigaHertz (GHz) e 16 GigaBytes (GB) de memória RAM.

[5]https://fasttext.cc
[6]https://www.tensorflow.org/
[7]https://keras.io
[8]http://www.cs.waikato.ac.nz/ml/weka/

3.2 ETAPAS DE PRÉ-PROCESSAMENTO

3.2.1 Tratamento do corpus

A primeira etapa desenvolvida no trabalho foi a de tratamento do corpus de mensagens. Inicialmente, balanceou-se o dataset para que o número de mensagens *spam* e *ham* fossem iguais. Como a proporção de mensagens é de 4.854 *ham* e 746 *spam*, existe o risco do classificador se viciar na classe majoritária, assim o *corpus* foi reduzido para 1.492 mensagens, sendo 746 de cada classe. Além disso, para garantir os resultados da aplicação das palavras nas RNAs e no Weka, dos dois conjuntos de mensagens (contendo 746 de cada classe), retirou-se 100 mensagens arbitrárias de cada um, para que estas formassem o conjunto de testes do trabalho. Assim, os conjuntos de dados do trabalho ficaram separados em:

- Conjunto A_1: utilizado para treinamento, consistindo de mensagens inteiras, contendo 1.292 mensagens com 646 de cada classe;
- Conjunto A_2: utilizado para teste e construído de forma semelhante ao conjunto A1, porém com 200 mensagens (100 de cada classe).

Em seguida, iniciou-se o processamento das mensagens, afim de obter as probabilidades das palavras do *corpus* estarem presentes em cada uma das classes de mensagens. O primeiro passo foi o de importar o *corpus*, originalmente formatado em um arquivo do tipo .csv (*comma separated values* – valores separados por vírgula, sucedendo a etiqueta de classe da sua mensagem correspondente). Neste passo, foi utilizanda a biblioteca pandas e o *corpus* foi inserido em uma estrutura de DataFrame [9] A estrutura obtida pode ser observada na Tabela 1.

Estando o corpus organizado na estrutura DataFrame, o próximo passo foi filtrar as mensagens e recuperar delas as palavras mais significantes, para isso, primeiramente foram removidas as pontuações e símbolos, em seguida, todas palavras foram transformadas para conter apenas letras minúsculas, e então, importando a lista de *stop words* presente na biblioteca nltk, foram removidas as palavras que não agregam significado às mensagens.

Com as palavras desejadas adquiridas, foram construídos mais dois conjuntos:

[9] DataFrame é uma estrutura bidimensional que facilita a manipulação de dados etiquetados, sua forma se assemelha a de uma tabela. É disponibilizada pela biblioteca pandas.

Tabela 1 – Mensagens organizadas na estrutura de DataFrame.

Rótulo	Mensagem
0	“Fine if thats the way u feel. Thats the way its gota b”
0	“I’m leaving my house now...”
1	“100 dating service cal;l 09064012103 box334sk38ch”
0	“Yes:)from last week itself i’m taking live call.”
1	“URGENT! Your Mobile number has been awarded with a £2000 prize [...]”
0	“Sorry,in meeting I’ll call later”
1	“your mobile number has won £5000, to claim calls us back[...]”
0	“I am great! How are you?”
1	“Talk sexy!! Make new friends or fall in love [...]”
1	“We know someone who you know that fancies you. Call 09058097218 to find out who.”

Fonte: Adaptado de Almeida et al. (2013).

- Conjunto B_1: contendo de modo avulso todas as palavras filtradas de A_1, onde cada palavra é repetida conforme a sua frequência em A_1;
- Conjunto B_2: contendo de modo avulso todas as palavras filtradas de A_2, exceto aquelas que já aparecem em B_1.

Além disso, elas foram categorizadas em três listas: (a) contendo todas as palavras do DataFrame (sem frequências); (b) contendo todas as palavras presentes nas mensagens spam e suas respectivas frequências; (c) contendo todas as palavras presentes nas mensagens ham e suas respectivas frequências. Essa separação foi feita com o intuito de poder realizar a contagem de palavras nessas três categorias, permitindo então obter os dados numéricos necessários para realizar os cálculos de atribuição das probabilidades às palavras do corpus.

3.2.2 Cálculo das probabilidades das palavras

Em posse das listas, utilizou-se a classe Counter da linguagem Python para fazer a contagem do total de palavras presentes no corpus (lista (a)), ou seja, o vocabulário do corpus, e para contabilizar a frequencia de cada palavra do vocabulário dentro das listas de *spam* (lista (b)) e *ham* (lista (c)), removendo duplicações e agrupando as palavras dessas listas. Essa biblioteca permite mapear as palavras em um dicionário[10] na forma relacional {chave: valor}, onde é possível recuperar informações como a frequência de cada palavra no dicionário (zero, caso a palavra não seja encontrada), as palavras mais frequentes, e tamanho do dicionário através dos métodos já presentes na biblioteca.

[10]Por dicionário, refere-se a estrutura baseada em Hash da linguagem Python. A lista de palavras do corpus é referenciada pelo termo vocabulário.

O algoritmo bayesiano ingênuo, em sua forma mais simples, realiza o cálculo da probabilidade das palavras da seguinte maneira: Seja w uma palavra (por exemplo, “promoção”) e c a classe de interesse (por exemplo, “spam”), o cálculo é realizado com base na frequência da palavra $f_c(w)$ em relação ao total de palavras do total de palavras na classe, denominado s_c. O cálculo é apresentado na equação 16:

$$p(w|c) = \frac{f_c(w)}{s_c} \tag{16}$$

No entanto, certas palavras podem não aparecer em todas as classes, logo, a probabilidade desta palavra será zero. Isto implica em anular o cálculo de probabilidade quando aplicado em uma mensagem, independente de quão expressivas sejam as probabilidades das outras palavras. Para contornar este problema, foi utilizada a suavização de Laplace. Esta técnica consiste em alterar a probabilidade de cada palavra assumindo que ela apareceu uma vez a mais do que sua real frequência em cada classe, somando o valor 1 ao numerador ($f(w)+1$), e fazendo o balanceamento levando em conta todas as possíveis palavras do vocabulário V, somando o tamanho de V ao denominador ($s_c + |V|$), resultando na equação 17.(CHEN; GOODMAN, 1996).

$$p(w|c) = \frac{f_c(w)+1}{s_c + |V|} \tag{17}$$

Aplicando a equação 17 sobre vocabulário V oriundo da lista (a) e sobre as frequências f_c oriundas das listas (b) e (c), foi possível calcular as probabilidades *spam* e *ham* de cada palavra do *corpus* e armazená-las em um arquivo de referência, aqui chamado de $P_{palavras}$.

Na Tabela 2 pode-se observar um exemplo da estrutura em que o arquivo $P_{palavras}$ está disposto. Nota-se que as probabilidades aqui são expressadas por números negativos, isto se deve ao fato de que os valores obtidos pelos cálculos foram muito pequenos, então apenas para que fossem melhor visualizados, aplicou-se sobre eles o logaritmo de base 10.

Tabela 2 – Palavras e probabilidades armazenadas em $P_{palavras}$

Palavra	Probabilidade Spam	Probabilidade Ham
ok	-3.861	-2.765
free	-2.167	-3.522
number	-1.206	-2.221
final	-3.317	-4.367
...	...	...
text	-2.438	-3.589
tkts	-3.986	-4.367
wkly	-3.384	-4.367

Fonte: Autoria própria.

Também é importante apontar o caso das duas últimas linhas, onde as palavras recuperadas das mensagens consistem em gírias (“tkts” para *tickets* - “ingressos”, em inglês) e (“wkly” para *weekly* - “semanalmente”, em inglês). Estas palavras, assim como outras presentes nas mensagens podem influenciar na qualidade das previsões, visto que podem gerar *embeddings* são muito distantes das demais palavras (escritas corretamente) do *corpus*.

3.2.3 Obtenção dos Word Embeddings

Para treinar o vocabulário em uma RNA, juntamente com as probabilidades adquiridas, a seguinte etapa foi a de buscar os *embeddings* de suas palavras, obtendo uma representação numérica e semântica sobre cada uma delas.

A fonte de *embeddings* utilizada consistiu na adoção de um modelo pronto que disponibiliza os arquivos de referência já treinados. Neste trabalho, foi utilizado o modelo disponibilizado por Bojanowski et al. (2016) contendo vetores de 300 dimensões extraídos e treinados a partir da Wikipedia da língua inglesa.

O treinamento das palavras é realizado executando a biblioteca fastText, que pode receber como argumentos a base de palavras, o modelo de aprendizado e a dimensão dos *embeddings* resultantes. O modelo escolhido foi o SkipGram e os vetores gerados por este modelo são compostos de 300 posições.

Após o treinamento do modelo, é gerado um arquivo de referência contendo duas informações por palavra: sua representação textual e seu respectivo vetor de *embedding*, como pode ser observado no exemplo da Tabela 3.

Tabela 3 – Palavras e seus embeddings

Palavra	**embedding[0]**	**embedding[1]**	**embedding[2]**	**embedding[3]**	**...**	**embedding[300]**
free	0.11583	-0.053446	-0.34493	0.46961	...	-0.34978
number	0.068904	0.047132	0.22284	0.42215	...	-0.060155
final	-0.00030806	0.29927	-0.081959	-0.14116	...	-0.12326
text	0.098738	0.60458	-0.3227	0.58527	...	-0.069712
receive	0.32148	-0.91803	-0.15973	0.25894	...	0.21543
hey	0.22232	0.54454	-0.6	0.70335	...	0.027114
word	0.22045	0.18056	0.35726	0.88367	...	-0.20542
back	0.5229	0.41538	-0.34014	0.29808	...	0.038043
like	-0.057102	-0.015598	-0.45095	0.65568	...	-0.30315

Fonte: Autoria própria.

4 EXPERIMENTOS

4.1 BAYESIANO INGÊNUO UTILIZANDO O WEKA

Para a realização desse experimento, os conjuntos A_1 e A_2 foram utilizados no software Weka como treinamento e teste, respectivamente. O algoritmo utilizado foi o NaiveBayesMultinomialText [1].

A aplicação do algoritmo na ferramenta Weka permite que alguns parâmetros sejam alterados para ajustar o treinamento, assim é possível alcançar melhores resultados sobre os testes. Os parâmetros que tiveram mais impacto no treinamento para a melhoria da acurácia foram os seguintes:

- useWordFrequencies: o modo de contabilizar as palavras na matriz de palavras (*bag of words*), seja com valores booleanos (verdadeiro para palavras que apareceram na mensagem) ou com valores numéricos (frequência individual da palavra). O padrão é usar valores booloeanos;
- minWordFrequency: a frequência mínima de palavras a ser considerada. O padrão é três;
- lowercaseTokens: a capitalização das palavras, no formato original ou convertendo para minúsculas. O padrão é manter a captialização.

Todos os demais parâmetros foram mantidos em seus valores padrão da versão 3.8.0 do Weka.

Uma característica importante a se ressaltar sobre o funcionamento do algoritmo NaiveBayesMultinomialText, é a de que se uma palavra no conjunto de testes não foi vista no conjunto de treinamento, ela é ignorada na realização do cálculo da multiplicação das probabilidades, mesmo que esta palavra tenha semântica semelhante a de alguma palavra do treinamento.

[1] http://weka.sourceforge.net/doc.dev/weka/classifiers/bayes/NaiveBayesMultinomialText.html

4.2 VALIDAÇÃO DAS PROBABILIDADES DAS PALAVRAS

O segundo experimento realizado neste trabalho compreendeu em verificar se as probabilidades obtidas pela etapa descrita na Seção 4.1 do pré-processamento das palavras são consistentes, para que então sejam utilizadas no treinamento dos modelos de RNA propostos para o trabalho.

Para realizar esta verificação, utilizou-se do arquivo $P_{palavras}$ apenas as palavras presentes em A_1 como referência para busca das probabilidades, e para teste, utilizou-se as mensagens presentes no conjunto A_2.

A verificação consiste em prever a classe de cada mensagem no conjunto A_2 com base nas probabilidades presentes no arquivo $P_{palavras}$, seguindo os seguintes passos: para cada mensagem, seleciona-se suas palavras válidas, do mesmo modo como realizado na Seção 3.3.1. Então, para cada palavra válida, verifica-se se ela está presente em $P_{palavras}$. Em caso positivo, soma-se o logaritmo da probabilidade *spam* e *ham* desta palavra em um acumulador dedicado a cada classe. Quando todas palavras válidas da mensagem forem verificadas, assinala-se à mensagem a classe referente ao acumulador que obteve a maior soma.

4.3 CRIAÇÃO E TREINAMENTO DAS RNAS

A realização deste experimento consiste na criação de duas RNAs. A entrada dessas redes são embeddings de palavras e suas respectivas saídas são as probabilidades conforme estimadas pelo algoritmo Bayesiano Ingênuo clássico. As RNAs são as seguintes:

- RNA_{spam}: especializada em aprender a probabilidade de cada palavra de B_1 ser *spam*;
- RNA_{ham}: semelhante a anterior, especializada em aprender a probabilidade de cada palavra de B_1 ser *ham*.

Partindo desta ideia, foram propostos modelos de topologias para avaliar o aprendizado as RNAs. Os modelos são redes multicamada completamente conectadas do tipo *feedforward*. São eles:

- Modelo M_1: Composto por três camadas de neurônios artificiais, sendo elas: (a) camada de entrada com 300 neurônios; (b) camada intermediária com 50 neurônios; (c) camada

de saída com 1 neurônio;

- Modelo M_2: Composto por três camadas de neurônios artificiais, sendo elas: (a) camada de entrada com 300 neurônios; (b) camada intermediária com 100 neurônios; (c) camada de saída com 1 neurônio;
- Modelo M_3: Composto por três camadas de neurônios artificiais, sendo elas: (a) camada de entrada com 300 neurônios; (b) camada intermediária com 200 neurônios; (c) camada de saída com 1 neurônio;
- Modelo M_4: Composto por quatro camadas de neurônios artificiais, sendo elas: (a) camada de entrada com 300 neurônios; (b) camada intermediária com 50 neurônios; (c) camada intermediária com 50 neurônios; (d) camada de saída com 1 neurônio;
- Modelo M_5: Composto por quatro camadas de neurônios artificiais, sendo elas: (a) camada de entrada com 300 neurônios; (b) camada intermediária com 100 neurônios; (c) camada intermediária com 50 neurônios; (d) camada de saída com 1 neurônio;
- Modelo M_6: Composto por quatro camadas de neurônios artificiais, sendo elas: (a) camada de entrada com 300 neurônios; (b) camada intermediária com 200 neurônios; (c) camada intermediária com 50 neurônios; (d) camada de saída com 1 neurônio.

A realização da montagem dos modelos foi feita utilizando os métodos de programação fornecidos pela biblioteca Keras. Utilizou-se as seguintes estruturas para iniciar a montagem dos modelos:

- Modelo de RNA *Sequential*: onde as camadas das RNAs são distribuídas de maneira enfileirada;
- Camadas do tipo *Dense*: onde todos neurônios estão conectados com a camada anterior, e sua ativação se dá a partir da definição dada na seção 2.3.1.

Para cada modelo, suas definições seguem o mesmo padrão de atributos: a função de ativação escolhida para as camadas de entrada e intermediárias foi a função Rectifier (ReLU) e os números de épocas escolhido para treinamento foram de 500, 1.000 e 2.000 épocas.

O modo de aprendizado pretendido com as RNAs é a busca de um modelo que possa relacionar o valor numérico das probabilidades de cada palavra com os seus *embeddings*, este problema se caracteriza como uma regressão. Por este motivo, a camada de saída é ativada pela função Linear, a função de custo a ser minimizada é a de erros quadráticos médios, e para otimização do aprendizado, utilizou-se o algoritmo Adam (KINGMA; BA, 2014).

Assim que o treinamento dos modelos das RNAs for concluído, a estratégia para a realização dos testes foi definida da seguinte maneira: para prever as probabilidades das palavras de B_2, utilizando-se das características vetoriais proporcionadas pelos *embeddings*, pode-se calcular sua distância Euclidiana (STEINBRUCH; WINTERLE, 2004) entre cada palavra de

B_2 em relação a B_1 e a partir da menor distância encontrada, ordená-las de maneira crescente, a fim de validar o quão capazes sãos as RNAs de reconhecer os *embeddings* e suas probabilidades conforme a proximidade das palavras.

4.4 AVALIAÇÃO DAS RNAS SOBRE AS MENSAGENS DO CORPUS

Este experimento consiste em escolher o melhor modelo de RNA_{spam} e o melhor modelo de RNA_{ham} dentre aqueles propostos no experimento 4.3, baseado nas suas medidas de correlação, e utilizá-los para predizer as classes de *spam* e *ham* das mensagens presentes no conjunto de testes A_2.

Para isto, foram criados seis conjuntos de palavras para avaliar os modelos:

- Conjunto A_1: contém todas as palavras filtradas do conjunto A_1. Este conjunto serve de base para verificar se as RNAs são capazes de realizar predições que possam ultrapassar os resultados obtidos no experimento 4.1;
- Conjunto A_{1+10}: contém todas as palavras filtradas do conjunto A_1 adicionando a ele as 10 palavras de A_2 mais próximas a A_1;
- Conjunto A_{1+20}: contém todas as palavras filtradas do conjunto A_1 adicionando a ele as 20 palavras de A_2 mais próximas a A_1;
- Conjunto A_{1+30}: contém todas as palavras filtradas do conjunto A_1 adicionando a ele as 30 palavras de A_2 mais próximas a A_1;
- Conjunto A_{1+50}: contém todas as palavras filtradas do conjunto A_1 adicionando a ele as 50 palavras de A_2 mais próximas a A_1;
- Conjunto A_{1+100}: contém todas as palavras filtradas do conjunto A_1 adicionando a ele as 100 palavras de A_2 mais próximas a A_1;
- Conjunto $A_1 + A_2$: contém todas as palavras filtradas do conjunto A_1 adicionando a ele todas as palavras de A_2;
- Conjunto A_2: contém todas as palavras do conjunto A_2. Este conjunto serve para analisar o poder de generalização da RNA .

A predição das mensagens é dada da seguinte forma: para cada mensagem, é realizado um filtro para selecionar todas suas palavras válidas. Em seguida, para cada palavra válida, verifica-se se ela está presente no conjunto de palavras avaliado. Se a palavra estiver contida no conjunto avaliado, seu *embedding* é introduzido em ambos modelos, para cada um calcular

sua respectiva probabilidade. Para cada palavra na mensagem, sua probabilidade *spam* e *ham* é somada a um acumulador dedicado a cada classe. Ao calcular a probabilidade para todas as palavras de uma mensagem, esta é classificada conforme o acumulador que obteve maior soma.

5 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados e discutidos os resultados obtidos na realização dos experimentos e testes definidos no capítulo 4 deste trabalho.

5.1 EXPERIMENTO COM O BAYESIANO IGNÊNUO UTILIZANDO O WEKA

Na Tabela 4, pode-se observar as combinações de parâmetros e a variação da acurácia do algoritmo NaiveBayesMultinomialText.

Tabela 4 – Experimentos com algoirtmo NaiveBayesMultinomialText

Experimento	minWordFrequency	useWordFrequencies	lowercaseTokens	Acurácia	Medida-F	Precisão	Cobertura
1	3	Falso	Falso	94%	0.94	0.944	0.94
2	3	Verdadeiro	Falso	95%	0.95	0.953	0.95
3	3	Falso	Verdadeiro	96,5%	0.965	0.967	0.965
4	3	Verdadeiro	Verdadeiro	97%	0.97	0.972	0.97
5	5	Falso	Falso	93.5%	0.935	0.937	0.935
6	5	Verdadeiro	Falso	93.5%	0.935	0.937	0.935
7	5	Falso	Verdadeiro	97%	0.97	0.971	0.97
8	5	Verdadeiro	Verdadeiro	97%	0.97	0.971	0.97
9	15	Falso	Falso	89,5%	0.895	0.9	0.895
10	15	Verdadeiro	Falso	89%	0.89	0.896	0.89
11	15	Falso	Verdadeiro	95%	0.95	0.951	0.95
12	15	Verdadeiro	Verdadeiro	94%	0.94	0.942	0.94
13	1	Falso	Falso	96,5%	0.965	0.965	0.965
14	1	Verdadeiro	Falso	97%	0.97	0.97	0.97
15	1	Falso	Verdadeiro	97,5%	0.975	0.975	0.975
16	1	Verdadeiro	Verdadeiro	98%	0.98	0.98	0.98

Fonte: Autoria própria.

Analisando o experimento 1 da Tabela 4, este traz os resultados para quando parâmetros utilizados pelo Weka estão em seus valores padrão. Nele percebe-se que o *software* considera que palavras com menos de 3 aparições no *corpus* são descartadas pelo o algoritmo. Nota-se também que pelo parâmetro **lowercaseTokens** ser inicialmente falso, pode-se entender que é feita uma distinção entre a capitalização das palavras. Suas medidas trazem um resultado

balanceado em comparação aos outros experimentos, ficando aproximadamente na média dos resultados (0.95).

Seguindo com os experimentos, alternou-se entre a relação da utilização ou não dos parâmetros **useWordFrequencies** e **lowercaseTokens** em relação à frequência mínima de cada palavra a ser considerada pelo parâmetro (**minWordFrequency**). É possível perceber a melhora é normalmente mais acentuada pelo uso do parâmetro **lowercaseTokens**, e que os melhores resultados são obtidos quando utiliza-se ambos parâmetros.

A Tabela 4 foi obtida a partir de uma busca em grade exaustiva no intervalo de 1 até 15 tokens de frequência mínima e valores verdadeiro e falso os parâmetros *useWordFrequencies* e *lowerCaseTokens*. Nota-se, porém, que somente os valores 1, 3, 5 e 15 para *minWordFrequency* foram incluídos na Tabela 4 devido ao fato de que os demais valores não trouxeram diferenças notáveis entre as medidas.

Nota-se que os resultados são melhores quando a frequência mínima de palavras a ser considerada pelo algoritmo vai diminuindo, podendo-se concluir que mesmo palavras com pouca frequência no *corpus* podem contribuir de alguma forma para as medidas.

O experimento 16, mostra que ao considerar todas as palavras para o cálculo do algoritmo Bayesiano Ingênuo e realizar a padronização da capitalização das palavras o algoritmo alcança os melhores resultados para o *corpus* de treinamento. Este é o experimento que servirá de referência para comparar os resultados obtidos pelas previsões das RNAs.

5.2 EXPERIMENTO PARA VALIDAÇÃO DAS PROBABILIDADES CALCULADAS

A Tabela 5 apresenta o resultado da avaliação realizada sobre as probabilidades calculadas durante a etapa de pré-processamento das mensagens de prever a classe das mensagens presentes no conjunto de testes A_2. Esse é importante para verificar se a implementação do Baysiano Ingênuo do Weka é equivalente a implementação do mesmo algoritmo neste trabalho, com a finalidade de poder comparar seus resultados.

O experimento da Tabela 4 na Seção 5.1 mais similar ao realizado nesta seção foi o experimento 16, devido a seus parâmetros. Observa-se, contudo, que para valores semelhantes de parâmetros, o Weka encontrou uma acurácia de 98,% conjuntamente para *spam* e *ham*. Na implementação do trabalho, foi obtido em média 95,4% em média para as duas classes. Há uma discrepância de cerca de 2,4 pontos percentuais, o que indica que internamente o Weka

Tabela 5 – Avaliação dos resultados obtidos pelas probabilidades calculadas.

Classe	Acurácia	Medida-F	Precisão	Cobertura
spam	0,95609	0,95609	0,933333	0,98
ham	0,95336	0,95336	0,97872	0,92929

Fonte: Autoria própria.

possui alguma variância na maneira que extrai as probabilidades de palavras ou que aplica o cálculo de probabilidade do Baysiano Ingênuo. Com esse resultado, fica mais difícil comparar a implementação deste trabalho com a implementação do Weka.

5.3 EXPERIMENTO DE TREINAMENTO DAS RNAS

Na Tabela 6, são apresentadas as correlações totais calculadas pela medida RSQ para os logaritmos das probabilidades calculadas pelos modelos propostos para as RNAs quando estes passaram por 500 épocas de treinamento.

Tabela 6 – Medida RSQ obtidos com o treinamento de 500 épocas para os modelos de RNA.

Modelo	Medida RSQ para a RNA Spam	Medida RSQ para a RNA Ham
M1	3.794×10^{-8}	0.01303796282
M2	0.0004999785939	0.01199077578
M3	0.00006259572213	0.01527550904
M4	0.0005433384745	0.0186495446
M5	0.0001106530616	0.02115328939
M6	0.002028732644	0.02450673248

Fonte: Autoria própria.

Nota-se que a medida demonstra que ambos *spam* e *ham* possuem baixas correlações, fenômeno mais acentuado em *spam*. Existem diferentes hipóteses para esse comportamento. A mais provável é que as palavras do conjunto A_2 se encontrem em regiões do espaço de palavras muito distantes das palavras contidas em A_1. Uma segunda hipótese é a presença de *underfitting* ou *overfitting* durante o treinamento da rede.

Observa-se ainda que os modelos M_4 para *spam* e M_6 para *ham* proporcionaram os melhores resultados no experimento, podendo ser um indicativo para a melhoria na estratégia

de criação de um novo modelo para as RNAs, baseado no número de neurônios das camadas intermediárias e na topologia contendo mais de uma camada intermediária.

Testes baseados em distância euclidiana ajudaram a investigar como a proximidade no espaço entre palavras de A_1 e A_2 influencia na capacidade de generalização das redes.

Na Figura 12(a) pode-se observar que quando treinados com apenas 500 épocas, os modelos da RNA_{spam} apresentam o mesmo padrão de queda abrupta de correlação das probabilidades, logo entre as 10 primeiras palavras, observando que as palavras no gráfico pertencem ao conjunto A_2 e são ordenadas por distância euclidiana de acordo com as palavras mais próximas no espaço que pertencem a A_1. Isso indica que as demais palavra de A_2 estão em regiões do espaço no qual a rede não consegue fazer boas generalizações.

De forma semelhante, a Figura 12(b) mostra que para 500 épocas também há uma queda grande na correlação das probabilidades conforme a distância das palavras verificadas pelos modelos de RNA_{ham} vai aumentando, embora que para estes, a correlação se mantém num nível mais alto por mais tempo, aproximadamente para 30 palavras mais próximas de A_1. Isso ajuda a explicar porque os modelos para *ham* se saíram melhores na Tabela 6.

As diferenças observadas entre os modelos para *spam* e para *ham* evidenciam uma distinção importante entre os dois tipos de mensagens. Mensagens de *spam* contém palavras muito raras, abreviações, gírias, URLs, entre outros. Além disso, em mensagens de *spam*, URLs são quebradas em textos menores separados por espaços, em uma tentativa de despistar sistemas automatizados para detecção de spam. O resultado, é que as palavras raras não ocupam boas regiões do espaço de *embeddings*. Já nas mensagens de *ham*, existem palavras muito mais comuns que ocupam regiões do espaço adequadas para o aprendizado das RNAs.

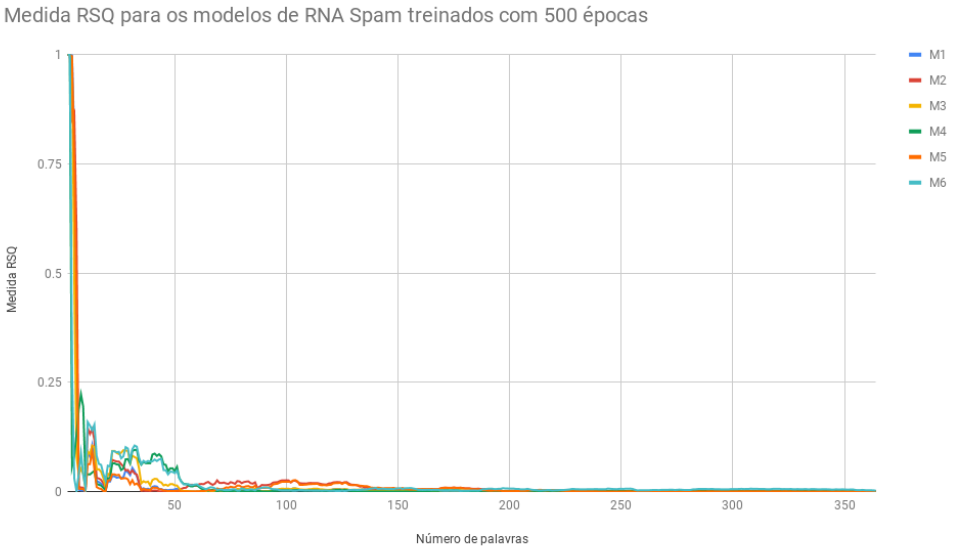
Para explorar a possibilidade da hipótese de haver *overfitting* nos modelos, decidiu-se observar o seus comportamentos ao passarem por apenas 100 épocas de treinamento.

Na Tabela 7, são mostradas as correlações calculadas pela medida RSQ para os logaritmos das probabilidades previstas pelos modelos quando estes tiveram 100 épocas de treinamento.

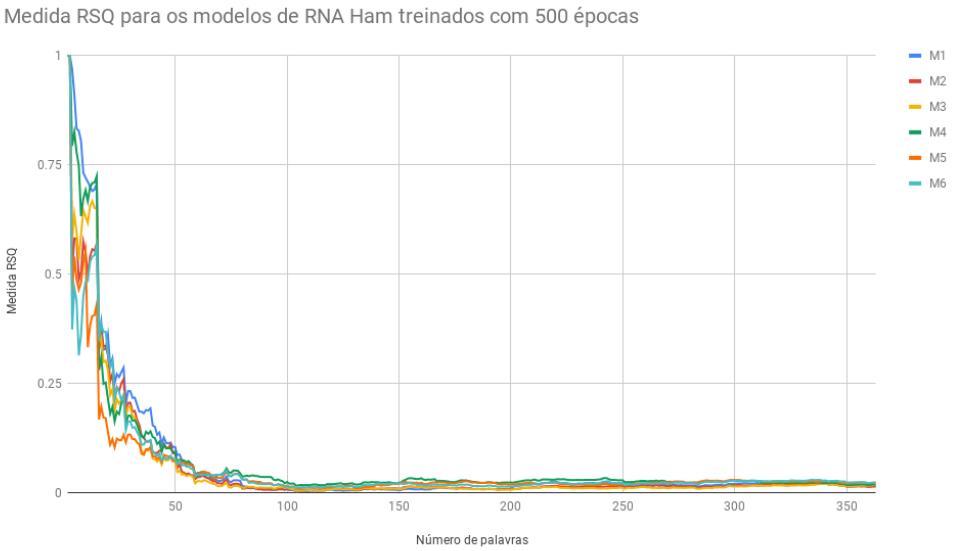
Tabela 7 – Medida RSQ obtidos com o treinamento de 100 épocas para os modelos de RNA.

Modelo	Medida RSQ para a RNA Spam	Medida RSQ para a RNA Ham
M1	0.0003347870469	0.01485014991
M2	0.00002097945293	0.02119154557
M3	0.000269640674	0.01055793126
M4	0.000269640674	0.0144779764
M5	0.0004195681002	0.02209437418
M6	0.0006056469746	0.02559778701

Fonte: Autoria própria.



(a)



(b)

Figura 12 – Medida RSQ obtidos com o treinamento de 500 épocas para os modelos de (a) RNA_{spam} **e (b)** RNA_{ham}**.**

Fonte: Autoria própria

Nota-se que os resultados obtidos realçam o contraste dos modelos, visto que de *ham* teve uma leve melhora e o de *spam* apresentou uma suave piora. A diferença entre os experimentos de 100 e 500 épocas são muito pequenas, não há grandes indícios que a rede treinada por 500 épocas tenha sofrido de *overfitting*. O próximo passo foi investigar a hipótese de *underfitting*, onde a alternativa para tentar melhorar os resultados é a de treinar os modelos

com mais épocas.

Na Figura 13(a) observa-se que a correlação para os modelos da RNA_{spam} treinados com 100 épocas, similar à do treinamento com 500 épocas, apresenta uma queda praticamente acentuada ao tentar prever as probabilidades *spam* das palavras do conjunto A_2, apontando que os modelos não foram capazes de fazer generalizações satisfatórias para as palavras ao diminuir o número de épocas.

Da mesma forma, a Figura 13(b) mostra que o comportamento dos modelos para a RNA_{ham} quando treinados com 100 épocas, apesar de ter atingido resultados parecidos, mesmo que pouco melhores que quando treinados com 500 épocas, ainda apresentam uma grande queda em sua correlação, novamente por volta de 30 palavras mais semelhantes a A_1. Para este experimento, nota-se que os modelos seguem um padrão muito semelhante no desenho de suas curvas, tendo apenas um pequeno deslocamento vinculado às suas correlações e com uma definição bem clara quanto ao local da queda da medida de correlação.

Continuando com a sequência de épocas propostas para treinamento, a Tabela 8 apresenta as medidas de correlação RSQ calculadas pelos modelos quando estes tiveram 1.000 épocas de treinamento.

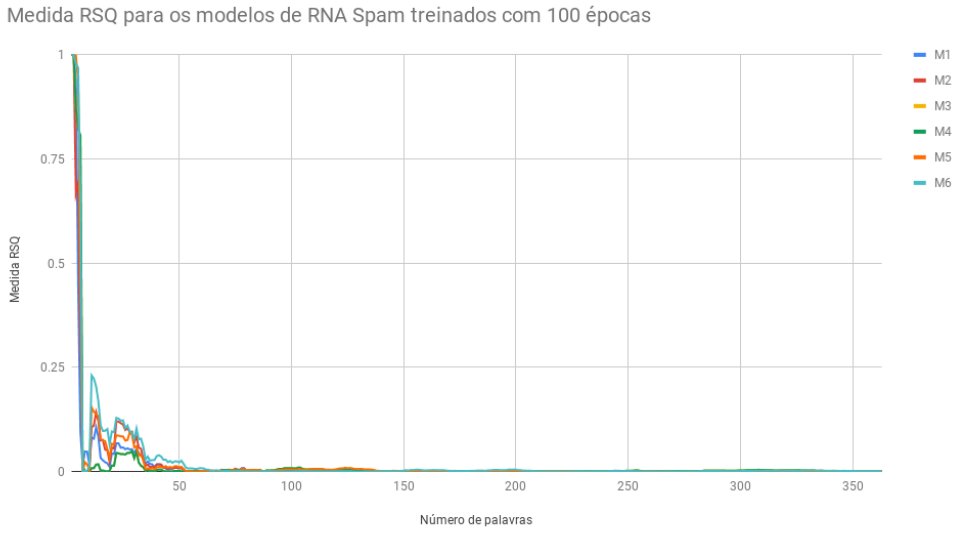
Tabela 8 – Medida RSQ obtidos com o treinamento de 1000 épocas para os modelos de RNA.

Modelo	Medida RSQ para a RNA Spam	Medida RSQ para a RNA Ham
M1	0.0001562631213	0.01041805451
M2	0.0003983757452	0.01530163179
M3	0.00009114271959	0.01195747717
M4	0.0004483383413	0.02234507992
M5	0.001768976372	0.01863300811
M6	0.002207755747	0.01222677331

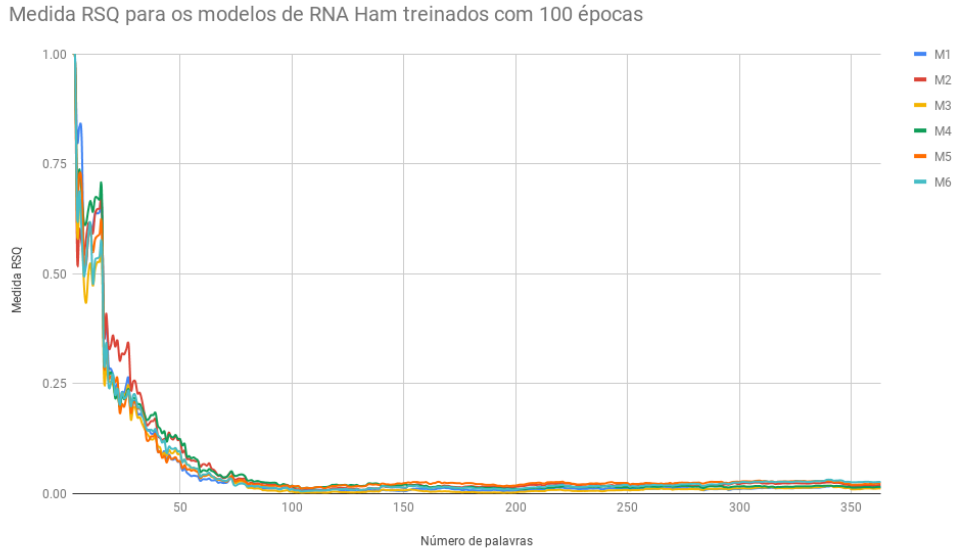
Fonte: Autoria própria.

Observa-se que o aumento para 1.000 épocas de treinamento dos modelos de RNA_{spam} não corresponde a um aumento na correlação dos modelos. Enquanto modelos como M_1, M_3, M_5 e M_6 tiveram pequenas melhorias em suas correlações, os modelos M_2 e M_4 obtiveram resultados piores. Já para os modelos de RNA_{ham} ocorre o oposto. Enquanto os modelos M_2 e M_4 melhoraram suas correlações, os modelos M_1, M_3, M_5 e M_6 tiveram uma redução em suas medidas. Embora que para todos os modelos em ambas RNA_{spam} e RNA_{ham}, a diferença em suas correlações em relação ao treinamento com 500 épocas é mínima.

A Figura 14(a) apresenta o gráfico da correlação dos modelos RNA_{spam} conforme sua previsão das palavras de A_2. Pode-se verificar que as curvas de correlação dos modelos mostram que praticamente não há melhorias em relação ao treinamento com 500 épocas, apresentado na



(a)



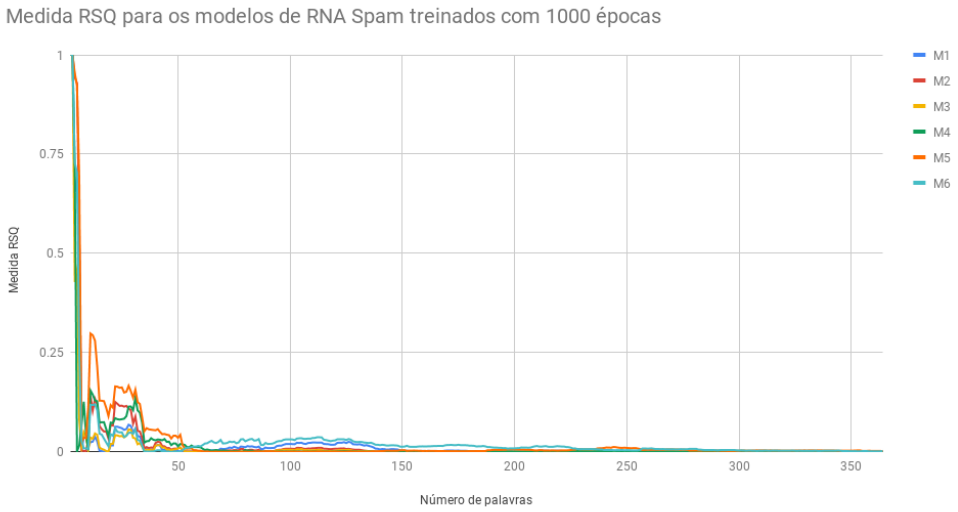
(b)

Figura 13 – Medida RSQ obtidos com o treinamento de 100 épocas para os modelos de (a) RNA_{spam} **e (b)** RNA_{ham}**.**

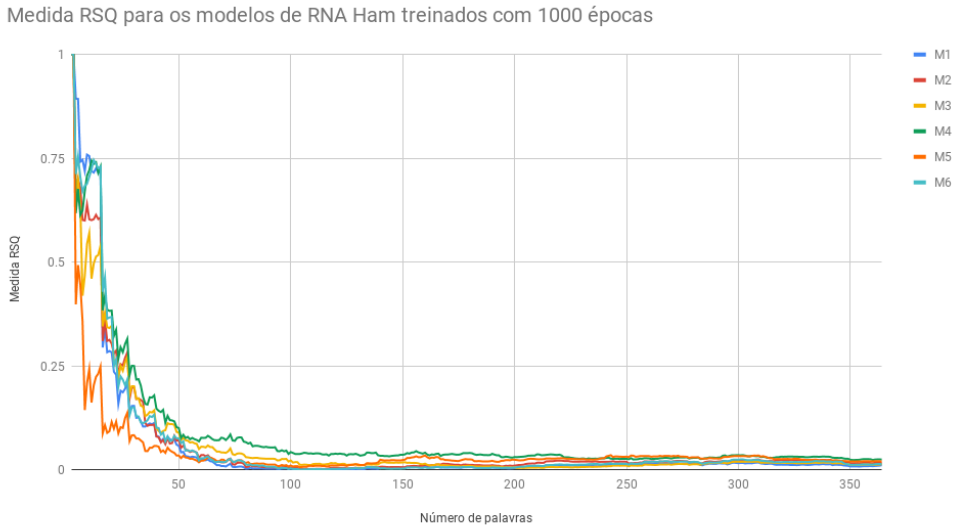
Fonte: Autoria própria

Figura 12(a). O único comportamento notável é um pico na medida de correlação do modelo M_5 ocorrido aproximadamente entre as 15 e 20 primeiras palavras, o que pode explicar a leve melhoria em sua medida em comparação ao seu treinamento com 500 épocas.

Em seguida, a Figura 14(b) permite observar que há uma queda ainda mais brusca na medida de correlação para quando os modelos RNA_{ham} são treinados com 1.000 épocas. Comparando suas curvas em relação aos treinamentos com 100 e 500 épocas (Figuras 13(b) e



(a)



(b)

Figura 14 – Medida RSQ obtidos com o treinamento de 1000 épocas para os modelos de (a) RNA_{spam} **e (b)** RNA_{ham}**.**

Fonte: Autoria própria

12(b)), percebe-se que há uma grande oscilação na medida logo nas primeiras palavras. Nota-se que o modelo M_5 também apresenta um comportamento bem diferente em relação aos outros modelos, indicando que a topologia de neurônios dada para M_5 se mostra ineficiente para este caso.

Por fim, observa-se na Tabela 9 que ao aumentar as épocas de treinamento para 2.000, os modelos não apresentaram mudanças que pudessem apontar melhora na correlação das predições das probabilidades das palavras.

Tabela 9 – Medida RSQ obtidos com o treinamento de 2.000 épocas para os modelos de RNA.

Modelo	Medida RSQ para a RNA Spam	Medida RSQ para a RNA Ham
M1	0.001502682033	0.005064509249
M2	0.0004990955186	0.02485813854
M3	0.00002389700061	0.009453023342
M4	0.0001376301877	0.03360496234
M5	0.00003803191172	0.01729365522
M6	0.0003195860784	0.01751297552

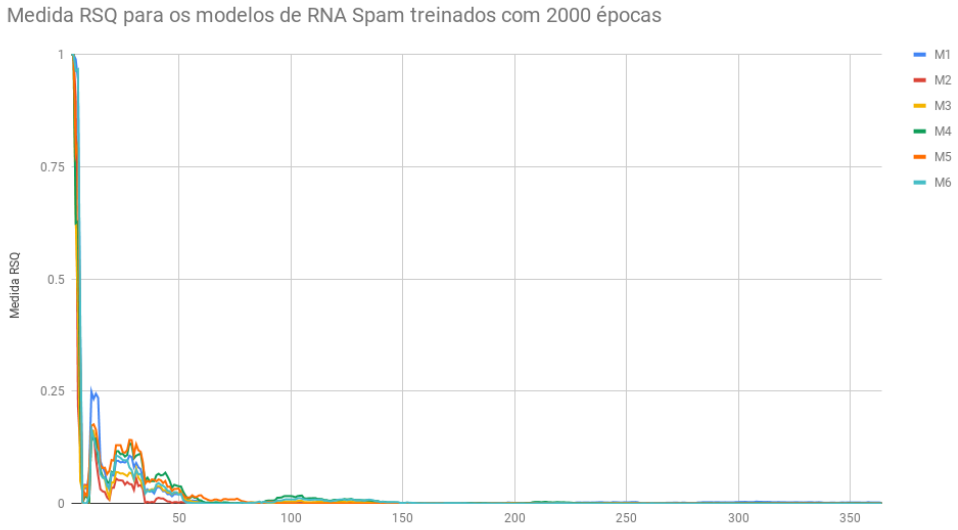
Fonte: Autoria própria.

Na Figura 15(a), as curvas referentes à medida de correlação calculada pelos modelos RNA_{spam} treinados com 2.000 épocas se mostram muito parecidas em relação aos modelos anteriormente treinados, mantendo consistente a imediata queda na correlação das palavras quando sua distância no espaço de palavras começa a crescer.

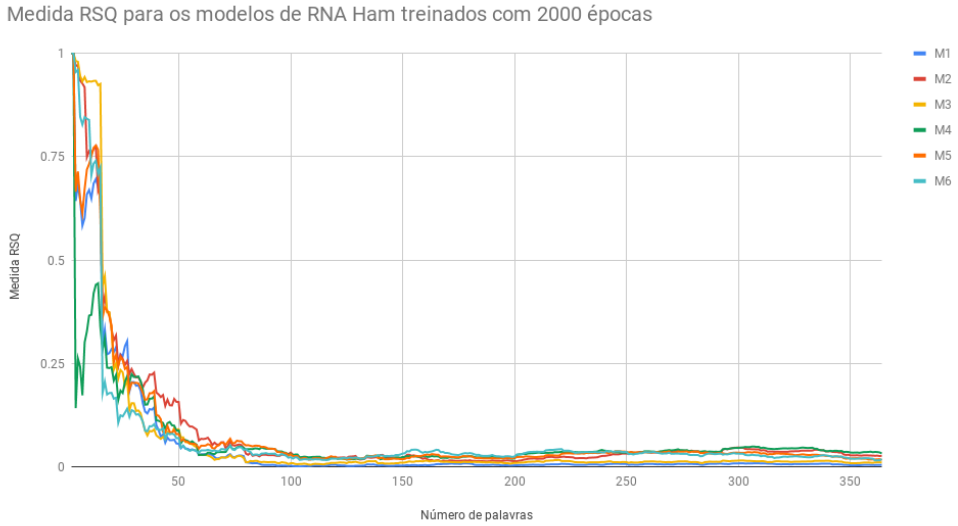
Seguindo para a Figura 15(b), pode-se observar que há um comportamento diferente para as primeiras palavras, aquelas com menores distâncias para o conjunto B_1. No treinamento com 2.000 épocas as correlações mantiveram-se com valores relativamente mais altos em comparação aos treinamentos passados, porém, novamente pelo fato das palavras presentes em B_2 apresentarem peculiaridades (abreviações, gírias, etc...) as medidas de correlação perdem a sua força conforme a distância vai aumentando, terminando em valores muito próximos aos obtidos nos experimentos anteriores.

Nestes experimentos pode-se constatar que os modelos propostos para predizer as probabilidades das palavras, mesmo dispondo de diferentes topologias e variadas épocas de treinamento, apresentaram resultados muito próximos e relativamente baixos. A hipótese para tal comportamento é a de que os dados extraídos do *corpus* utilizado neste trabalho são pequenos para permitir que a rede generalize para bem novas palavras. Além disso, o problema é pior na base de *spam* visto que existem gírias, abreviações e URLs que não permitem a geração de bons embeddings.

Para seguir ao experimento de avaliação das RNAs sobre as mensagens do corpus, levando em conta as medidas RSQ calculadas após as suas predições, os modelos escolhidos foram: (a) para RNA_{spam}, o modelo M_6 treinado com 1000 épocas; (b) para RNA_{ham}, o modelo M_4 treinado com 2.000 épocas.



(a)



(b)

Figura 15 – Medida RSQ obtidos com o treinamento de 2.000 épocas para os modelos de (a) RNA_{spam} **e (b)** RNA_{ham}**.**

Fonte: Autoria própria

5.4 EXPERIMENTO DE AVALIAÇÃO DAS RNAS SOBRE AS MENSAGENS DO CORPUS

Nas tabelas 10 e 11 observa-se as medidas de avaliação dos modelos de RNA para as mensagens por eles classificadas como *spam* em *ham*.

Tabela 10 – Avaliação dos modelos de RNA para mensagens *spam*.

Modelo	Acurácia	Medida-F	Precisão	Cobertura
A_1	0.95609	0.95609	0.933333	0.98
A_{1+10}	0.95609	0.95609	0.933333	0.98
A_{1+20}	0.95609	0.95609	0.933333	0.98
A_{1+30}	0.96078	0.96078	0.9423	0.98
A_{1+50}	0.96078	0.96078	0.9423	0.98
A_{1+100}	0.96078	0.96078	0.9423	0.98
$A_1 + A_2$	0.96585	0.96585	0.94285	0.99
A_2	0.73796	0.73796	0.793103	0.69

Fonte: Autoria própria.

Tabela 11 – Avaliação dos modelos de RNA para mensagens *ham*.

Modelo	Acurácia	Medida-F	Precisão	Cobertura
A_1	0.95336	0.95336	0.97872	0.92929
A_{1+10}	0.95336	0.95336	0.97872	0.92929
A_{1+20}	0.95336	0.95336	0.97872	0.92929
A_{1+30}	0.95876	0.95876	0.97894	0.93939
A_{1+50}	0.95876	0.95876	0.97894	0.93939
A_{1+100}	0.95876	0.95876	0.97894	0.93939
$A_1 + A_2$	0.96373	0.96373	0.98936	0.93939
A_2	0.76555	0.76555	0.72072	0.81632

Fonte: Autoria própria.

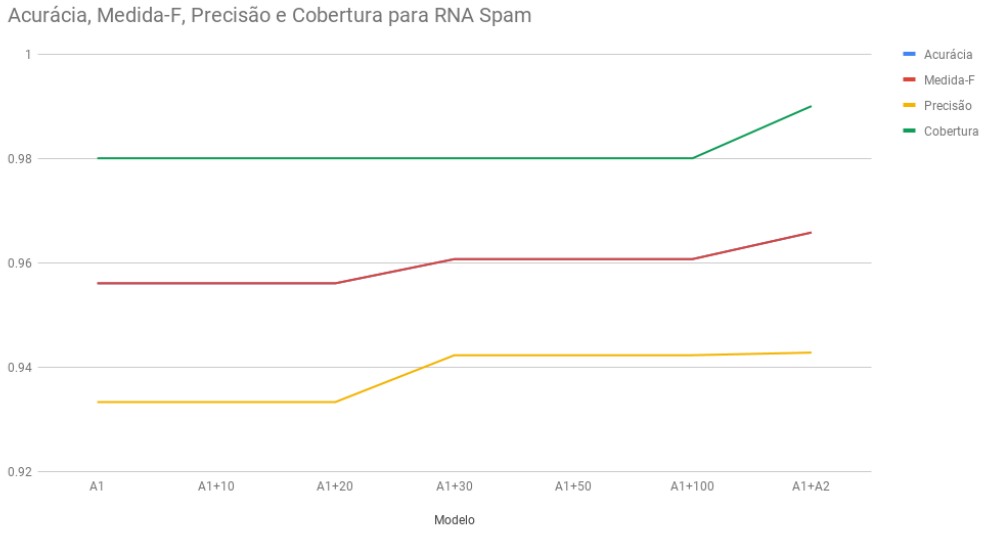
Os experimentos rotulados como A_1 nas Tabelas 10 e 11 são a base para os experimentos seguintes. Seu propósito é fazer uma avaliação extrínseca da rede para estimar probabilidades de palavras conhecidas, e permite fazer análises adicionais em relação a avaliação intrínseca baseada em correlação apresentada nos gráficos 14(a) e 15(b). Além disso, esses dois experimentos também permitem comparar como a rede se comporta em relação ao experimento mais semelhante utilizando a implementação do Weka (Experimento 16 da Tabela 4).

Ainda em relação aos experimentos rotulados como A_1 observa que são exatamente os mesmos valores apresentados na Tabela 5 da Seção 5.2. Isso indica que a rede está estimando muito bem as probabilidades das palavras conhecidas. Contudo, fica difícil o comparativo com o Weka (Experimento 16 da Tabela 4 da Seção 5.1) devido as razões discutidas na Seção 5.2.

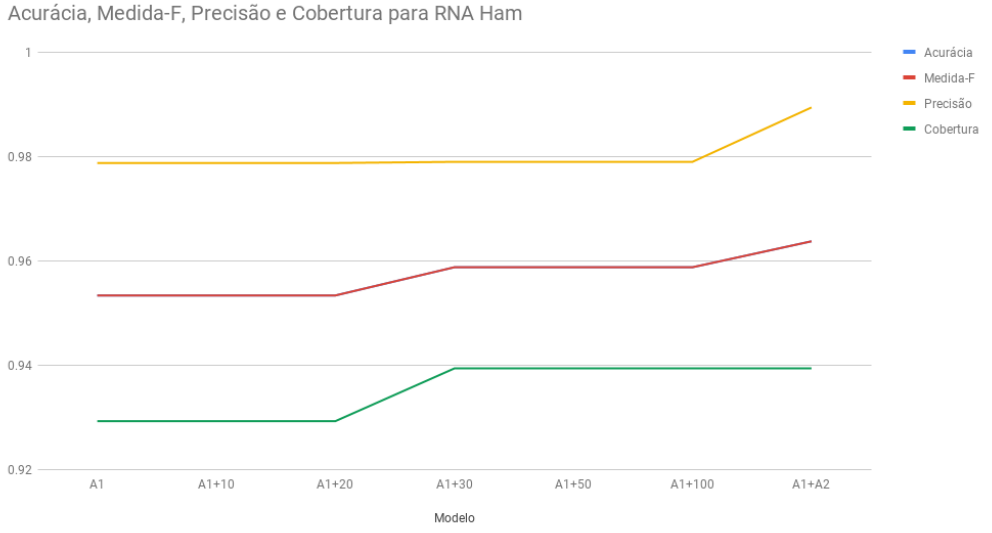
Esse comportamento é válido para todos os experimentos restantes das tabelas 10 e 11, mostrando que mesmo com as medidas de correlação nos testes dos modelos apresentando resultados baixos, as RNAs foram capazes de fazer uma generalização suficiente para que os *embeddings* possam ser utilizados para prever a classe das mensagens tão bem quanto a

configuração mais básica do algoritmo utilizado pelo Weka.

O resultado mais importante deste experimento é a constatação que há uma leve melhora no desempenho do algoritmo Baysiano Inguênuo proposto neste trabalho a medida que palavras desconhecidas vão sendo apresentadas às RNAs e suas probabilidades generalizadas vão sendo utilizadas. Esse comportamento pode ser observado na Figura 16. Por um lado, nota-se que a melhora é muito leve e que não foram feitos testes de significância estatística para verificar se a melhora é de fato autêntica. Por outro lado, este trabalho fez uso de uma base pequena, tanto no que se refere a quantidade de palavras quanto a quantidade de mensagens. Isso sugere que a utilização das RNAs é promissora para melhorar o desempenho do Baysiano Ingênuo.



(a)



(b)

Figura 16 – Medidas de avaliação das RNAs sobre as mensagens do *corpus*.

Fonte: Autoria própria

6 CONCLUSÕES

Neste trabalho, foi descrito o processo de utilização de uma base de mensagens SMS para efetuar o treinamento de RNAs capazes de classificar mensagens entre *spam* e *ham*, conforme a predição do logaritmo da probabilidade calculado pelas RNAs para os *embeddings* obtidos a partir das palavras consideradas válidas pelo pré-processamento de seu texto.

Como base para comparação dos resultados obtidos pelas previsões das RNAs, as mensagens do *corpus* foram introduzidas no software Weka. Foram realizados experimentos utilizando o seu algoritmo de previsão de probabilidades "NaiveBayesMuiltinomialText", onde foram alcançados resultados de grande precisão através de alterações em seus parâmetros.

Verificou-se alguma discrepância entre o algoritmo Bayesiano Ingênuo implementado no software Weka e o algoritmo implementado neste trabalho, observando-se que a implementação disponibilizada pelo Weka possui otimizações internas que se apresentaram superiores à abordagem do Bayesiano Ingênuo clássico desenvolvida neste trabalho. Porém com a implementação do Weka não foi possível extrair os dados necessários para a construção e utilização dos *embeddings* em conjunto com as RNAs, fazendo com que, mesmo que sua implementação seja superior, não foi possível utilizá-la em conjunto com os métodos abordados no trabalho.

Os modelos de RNAs para prever as probabilidades foram treinados a partir dos *embeddings* das palavras B_1 presentes no conjunto de mensagens A_1. buscou-se investigar o desempenho destes modelos quando estes foram usados para realizar previsões de palavras nunca antes vistas (as palavras do conjunto de testes B_2). Para obter uma avaliação onde fosse possível verificar a utilidade dos *embeddings*, estes foram ordenados de maneira crescente com base na sua distância euclidiana com relação às palavas do conjunto de treinamento, visando verificar a relação entre a qualidade da previsão e a disposição dos vetores no espaço de palavras.

Observando os resultados medidos através da medida de correlação RSQ, foi possível notar uma grande deficiência na previsão de probabilidades das RNAs, onde atribui-se este comportamento à suspeita de que pelo fato de boa parte das palavras contidas no conjunto de palavras B_2 presentes no conjunto de mensagens de *spam* A_2 serem gírias, abreviações, URLs e algumas sequências de caracteres sem muito sentido, os *embeddings* gerados para estas

palavras encontram-se em regiões do espaço de palavras muito distantes daquelas usadas como treinamento das RNAs, sendo assim, os modelos não conseguiram realizar boas generalizações para estas palavras.

O experimento de conclusão do trabalho foi o mais promissor, onde os melhores modelos escolhidos segundo sua medida de correlação no experimento anterior foram utilizados para prever as probabilidades e assinalar as classes de *spam* e *ham* às mensagens do conjunto de testes A_2 e classificar as mensagens em suas respectivas classes. Os modelos de RNAs demonstraram ser capazes de realizar previsões com resultados bastante satisfatórios. Verificou-se que, com a introdução de novas palavras, as RNAs permitiram que o Bayesiano Ingênuo apresentasse leve melhora nos seus resultados, embora testes estatísticos sejam necessários para confirmar essa tendência. De toda a forma, os resultados sugerem, que em grandes conjuntos de dados, as RNAs podem dar uma contribuição importante ao processo de detecção de *spam* e *ham* por meio do algoritmo Bayesiano Ingênuo.

Por fim, ao recapitular os objetivos deste trabalho, constata-se que todas as etapas foram concluídas e os resultados esperados foram alcançados, mostrando que as técnicas aqui apresentadas para filtragem de *spam* em mensagens SMS se mostraram promissoras.

6.1 TRABALHOS FUTUROS

Para trabalhos futuros, pode-se apontar:

- Verificar o desempenho das RNAs ao serem treinadas sobre as palavras uma base de e-mails, onde há uma maior quantidade de dados presentes para calcular probabilidades;
- Calcular probabilidades utilizando o algoritmo dos K-Vizinhos Mais próximos ao invés de redes neurais. Para tal, é preciso guardar-se as probabilidades de algumas palavras de *spam* e *ham*, e estima-se a probabilidade de novas palavras com base nas probabilidades de suas vizinhas mais próximas;
- Conduzir testes de hipótese para verificar se o Bayesiano Ingênuo baseado nas RNAs é significativamente melhor que o Bayesiano Ingênuo clássico.
- Utilizar outras medidas de distâncias tais como Manhattan e Geodésica.

REFERÊNCIAS

ALMEIDA, T. A.; HIDALGO, J. M. G.; SILVA, T. P. Towards SMS Spam Filtering: Results under a New Dataset. **International Journal of Information Security Science (IJISS)**, p. 1–18, 2013. Disponível em: <http://www.dt.fee.unicamp.br/˜tiago//smsspamcollection/IJISS13.pdf>.

BATISTA, G. E. de A. P. A. **Pré-processamento de Dados em Aprendizado de Máquina Supervisionado**. Doutorado, 2003.

BOJANOWSKI, P. et al. Enriching word vectors with subword information. **Transactions of the Association for Computational Linguistics**, v. 5, p. 135–146, 2016.

CHAUBARD, F.; MUNDRA, R.; SOCHER, R. **Deep Learning for NLP**. 2016. `http://cs224d.stanford.edu/lecture_notes/notes1.pdf`.

CHEN, S. F.; GOODMAN, J. An empirical study of smoothing techniques for language modeling. In: **Proceedings of the 34th Annual Meeting on Association for Computational Linguistics**. Association for Computational Linguistics, 1996. p. 310–318. Disponível em: <https://doi.org/10.3115/981863.981904>.

CHOWDHURY, G. G. Natural Language Processing. **Annual Review of Information Science and Technology**, v. 37, p. 51–89, 2003.

DHILLON, P. S.; FOSTER, D. P.; UNGAR, L. H. Eigenwords: Spectral Word Embeddings. **Journal of Machine Learning Research**, v. 16, p. 1–36, 2015. Disponível em: <http://www.pdhillon.com/dhillon15a.pdf>.

ERICSSON. 2018. Disponível em: <https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf>.

ESENDEX. 2018. Disponível em: <https://www.esendex.co.uk/blog/post/what-is-the-open-rate-for-sms-in-2018/>.

FERNANDES, A. M. da R. **Inteligência Artificial - Noções Gerais**. [S.l.: s.n.], 2003.

FINNOCCHIO, M. A. F. Noções de redes neurais artificiais. 2014.

GOLDBERG, Y. A Primer on Neural Network Models for Natural Language Processing. **Journal of Artificial Intelligence Research**, v. 57, p. 345–420, 2015. Disponível em: <http://arxiv.org/pdf/1510.00726v1.pdf>.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. [S.l.: s.n.], 2001.

HAYKIN, S. **Neural Networks and Learning Machines**. [S.l.: s.n.], 2009.

HIDALGO, J. M. G. et al. Content Based SMS Spam Filtering. **Proc. of the 2006 DOCENG**, p. 107–114, 2006. Disponível em: <http://portal.acm.org/citation.cfm?doid=1166160.1166191>.

JOYCE, J. Bayes' theorem. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Winter 2016. [S.l.]: Metaphysics Research Lab, Stanford University, 2016.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition**. [S.l.: s.n.], 2009.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **CoRR**, 2014. Disponível em: <http://arxiv.org/abs/1412.6980>.

LACERDA, W. S.; BRAGA, A. d. P. Experimento de um Classificador de Padrões Baseado na Regra Naive de Bayes. **InfoComp - Revista de Computação da UFLA**, p. 30–35, 2004.

LOUDHOUSE. 2014. Disponível em: <https://blogs.sap.com/2014/12/15/is-sms-still-a-wise-choice/>.

MIKOLOV, T. et al. **Distributed Representations of Words and Phrases and their Compositionality**. 2013. `http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf`.

MITCHELL, T. M. **Machine Learning**. [S.l.: s.n.], 1997.

MONARD, M. C.; BARANAUSKAS, J. A. **Conceitos sobre Aprendizado de Máquina**. [S.l.: s.n.], 2003. 89–114 p.

NUNES, M. D. G. V. O Processamento de Línguas Naturais: para quê e para quem? **Instituto de Ciências Matemáticas e de Computação**, v. 73, p. 12, 2008.

RISH, I. An empirical study of the naive Bayes classifier. **IJCAI 2001 workshop on empirical methods in artificial intelligence**, p. 41–46, 2001.

ROJAS, R. **Neural Networks: A Systematic Introduction**. [S.l.: s.n.], 1996.

SMSGLOBAL. 2018. Disponível em: <https://thehub.smsglobal.com/sms-marketing-2018>.

STEINBRUCH, A.; WINTERLE, P. **Geometria Analitica**. Pearson Makron Books, 2004. ISBN 9780074504093. Disponível em: <https://books.google.com.br/books?id=tOLfGwAACAAJ>.

ZHANG, H. Y.; WANG, W. Application of Bayesian Method to spam SMS filtering. **Proceedings - 2009 International Conference on Information Engineering and Computer Science, ICIECS 2009**, p. 1–3, 2009.

ZOGBY. 2014. Disponível em: <https://bit.ly/2OaVgVs>.