Leandro Massetti Ribeiro Oliveira

Inteligência Artificial Aplicada a Detecção de Fake News

São Luís - Maranhão

15 de Julho de 2019

Leandro Massetti Ribeiro Oliveira

Inteligência Artificial Aplicada a Detecção de Fake News

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia da Computação da UFMA, como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação, Centro de Ciência Exatas e Tecnológicas da Universidade Federal do Maranhão.

Universidade Federal do Maranhão

Centro de Ciências Exatas e Tecnologia

Engenharia da Computação

Orientadora: Vandecia Rejane Monteiro Fernandes

São Luís - Maranhão

15 de Julho de 2019

Leandro Massetti Ribeiro Oliveira

Inteligência Artificial Aplicada a Detecção de Fake News / Leandro Massetti Ribeiro Oliveira. – São Luís - Maranhão, 15 de Julho de 2019-

43 p. : il. (algumas color.) ; 30 cm.

Orientadora: Vandecia Rejane Monteiro Fernandes

Trabalho de Conclusão de Curso – Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Engenharia da Computação, 15 de Julho de 2019.

1. Fake News. 2. Aprendizado de Máquina. 3. Processamento de Linguagem Natural I. Vandecia Rejane Monteiro Fernandes. II. Universidade Federal do Maranhão. III. Inteligência Artificial Aplicada a Detecção de Fake News

Leandro Massetti Ribeiro Oliveira

Inteligência Artificial Aplicada a Detecção de Fake News

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia da Computação da UFMA, como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação, Centro de Ciência Exatas e Tecnológicas da Universidade Federal do Maranhão.

Trabalho aprovado. São Luís - Maranhão, 15 de Julho de 2019:

Vandecia Rejane Monteiro Fernandes
Orientadora

Bruno Feres de Souza
Universidade Federal do Maranhão

Alex Oliveira Barradas Filho
Universidade Federal do Maranhão

São Luís - Maranhão
15 de Julho de 2019

Resumo

Informações são veiculadas ao público através da internet, televisão, jornais, revistas e outros meios de comunicação a todo instante, mas nem sempre se apresentam com conteúdo confiável. O acesso a esse tipo de conteúdo e a propagação de notícias que não são verdadeiras torna-se um fato alarmante por suas intensões. As *Fake News* são conhecidas por essas características, o uso mal intencionado de informações pode comprometer a democracia, manipulando a opinião do leitor. Preocupado com a gravidade que o tema apresenta à nossa sociedade, este trabalho propôs a detecção de *Fake News*, em meio a notícias verdadeiras, utilizando Algoritmos de inteligência artificial como Árvore de Decisão, *Naive Bayes*, SVM e Aprendizagem Profunda. Para analisar esses padrões dentro de notícias assim foram utilizadas neste trabalho duas abordagens, A e B. A abordagem A foi utilizada para o uso dos algoritmos clássicos de aprendizagem de máquina, Árvore de Decisão, *Naive Bayes* e SVM, enquanto que a abordagem B utilizou as redes neurais. Foram escolhidas três redes, ainda na abordagem B, que mostraram melhores resultados, e foi verificado seu desempenho no conjunto de teste. Os resultados para abordagem A mostraram que o algoritmo SVM teve maior acurácia, atingindo 90,13%, e da abordagem B a rede 1 foi a melhor atingindo a acurácia de 92,36%.

palavra-chave: Fake News; Aprendizado de Máquina; Processamento de Linguagem Natural.

Abstract

Information is conveyed to the public through the internet, television, newspapers, magazines and other media at all times, but they do not always present themselves with reliable content. The access to this type of content and the spread of news that is not true becomes an alarming fact because of its intentions. Fake News are known for these characteristics, the misuse of information can compromise democracy, manipulating the opinion of the reader. Concerned about the seriousness of the theme in our society, this work proposed the detection of Fake News, between true news, using artificial intelligence algorithms such as Decision Tree, Naive Bayes, SVM and Deep Learning. In order to analyze these patterns within news, two approaches, A and B, were used in this paper. The A approach for the use of classical machine learning Algorithms, Decision Tree, Naive Bayes, SVM, while B approach used the deep neural networks. Three networks were chosen, still in B approach, that showed better results, and their performance in the test set was verified. The results for the A approach showed that the SVM algorithm was more accurate, reaching 90.13%, and from B approach the network 1 was the best reaching the accuracy of 92,36%.

Key-words: Fake News; Machine Learning; Natural Language Processing.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AM	Aprendizado de Máquina
DL	*Deep Learning*
FN	Falso-negativo
FP	Falso-positivo
GPU	Unidades de Processamento Gráficos
KDD	*Knowledge Discovery in Databases*
KDT	*Knowledge Discovery in Texts*
LSTM	*Long short-term memory*
MPL	*Multilayer Perceptron*
NLTK	*Natural Language Toolkit*
RBF	*Radial Basis Function*
RNN	*Recurrent neural network*
SRM	*Structural Risk* Minimization
SVM	*Support Vector Machine*
TAE	Teoria do Aprendizado Estatístico
TF-IDF	Term Frequency - Inverse Document Frequency
TVN	Taxa de Verdadeiros-negativos
TVP	Taxa de Verdadeiros-positivos
VN	Verdadeiros-negativos
VP	Verdadeiros-positivos

Sumário

1 Introdução

De acordo com o dicionário Aurélio (1986) notícia é a difusão de informação através da imprensa falada ou escrita, em outras palavras, é um acontecimento novo e recente ou que divulga uma novidade sobre uma situação já existente. Estas informações podem ser veiculadas ao público através da internet, televisão, jornais, revistas e outros meios de comunicação.

De modo geral uma notícia pode ser repassada por fontes oficiais onde a imparcialidade garante que a mesma seja exposta "tal como ela é", livre de opiniões ou adjetivos que as engrandeça ou as diminua perante a realidade dos fatos. No âmbito jornalístico, de acordo com Alberto (1970), a ética, que é o conjunto de normas e procedimentos éticos que regem a atividade do jornalista, garante que estas informações sejam repassadas sem ruídos.

Bem como as narrativas anteriormente descritas, que destacam-se pela veracidade de seus fatos, a clareza da linguagem e a objetividade do conteúdo, exitem também aquelas que são criadas com o intuito de manipular o leitor, essas são notícias incorretas, ou mesmo uma notícia que contém informação incorreta cercada de outras verdadeiras, notícias assim, de acordo com Jr, Lim e Ling (2018), são denominas *Fake News*.

Existe portanto uma necessidade de identificação das mesmas para que menos pessoas sejam prejudicadas por seus conteúdos. É possível fazer uma verificação em fontes confiáveis quanto a credibilidade do assunto, porém é um processo trabalhoso e que não vem demonstrando resultado devido ao crescente uso das mídias sociais para disseminação dessas notícias.

No entanto, existem características nas *Fake News* quanto ao seu padrão de escrita que as diferenciam de notícias verdadeiras. Essas características e padrões no texto podem ser uma ferramenta na identificação das mesmas. Uma alternativa para a identificação delas seria o uso de técnicas computacionais, como o processamento de linguagem natural.

Este trabalho objetiva aplicar técnicas de inteligência artificial e processamento de linguagem natural para identificação de *Fake News* entre notícias reais.

1.1 Objetivo Geral

Este trabalho objetiva a identificação automática de *Fake News* dentre notícias reais utilizando técnicas de inteligência artificial e processamento de linguagem natural.

1.2 Objetivos Específicos

- Utilizar algoritmos de aprendizado de máquina na detecção de *fake news*
- Utilizar a técnica de Aprendizado Profundo para detecção do problema.
- Comparar as metodologias utilizadas, analisando os resultados obtidos.

1.3 Organização do trabalho

O trabalho está organizado em cinco capítulos, de forma a apresentar o conteúdo mais claramente, conforme os parágrafos a seguir.

O Capítulo 2, Fundamentação Teórica, apresenta o embasamento teórico necessário para o entendimento e desenvolvimento da metodologia proposta.

O Capítulo 3, Metodologia, apresenta as etapas realizadas para o desenvolvimento do modelo proposto para a detecção de *fake news*.

O Capítulo 4, Resultados, apresenta e discute os resultados obtidos pela metodologia proposta.

O Capítulo 5, Conclusão, apresenta as considerações finais acerca do trabalho realizado.

2 Fundamentação Teórica

2.1 Fake News

As *Fake News*, ou notícias falsas, de acordo com Jr, Lim e Ling (2018), são notícias com conteúdo falso e sem embasamento, com o intuito de enganar e manipular o leitor. Elas atingem de maneira negativa a sociedade porque não são de fato apenas informações errôneas, mas também informações criadas com o intuito de enganar quem as recebe, tornando a prática oportuna a interessados que as disseminam, resultando em situações alarmantes em pequenos cenários ou até mesmo em esferas mundiais. As eleições estadunidense de 2016, para presidente, promoveram um cenário ímpar com relação as *Fake News* na internet, uma vez que o termo passou a ser utilizado com mais recorrência pela população e pela imprensa internacional, e se manteve em alta desde então.

Em sites que dizem respeito à criação de *Fake News*, de acordo com MARUMO (2018) algumas características usadas foram registros com domínio ".com"ou ".org", dificultando a identificação dos responsáveis com a mesma transparência que acontecem com os domínios registrados no Brasil, que terminam com ".br". Além disso, seus nomes são semelhantes a outros de jornais e de blogs autorais. Na parte do conteúdo do site as notícias não são assinadas e são cheias de opiniões, além de possuir um layout poluído e repleto de propagandas. Geralmente não possuem a página de "Quem Somos", e quando possuem não deixam claro quem são os autores do site (APRÁ, A, 2017).

Em contrapartida, no que se trata de disseminação de *Fake News*, Ruediger et al. (2017) relata que a automatização de ferramentas de publicação possibilitou o surgimento e a propagação de robôs — contas controladas por *softwares* se fazendo passar por seres humanos que já dominam parte da vida nas redes sociais e participam ativamente das discussões em momentos políticos de grande repercussão. Assim, temos um cenário propício a esse tipo de informação, a *internet*, e pessoas ou *softwares* fazendo o envio em massa das *Fake News*, com uso de mídias sociais ou até mesmo robôs.

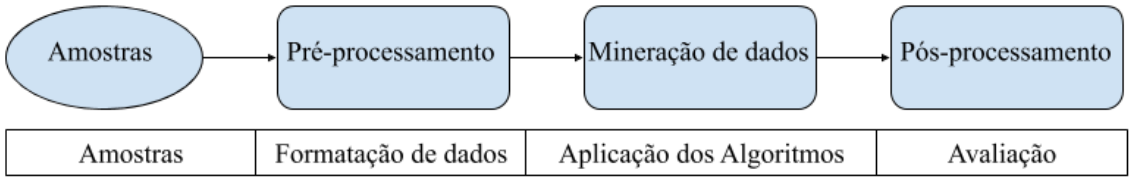
O trabalho de Granik e Mesyura (2017) destaca a semelhança entre *Fake News* e *spam*, que de acordo com contributors (2019) é o uso de sistemas de mensagens eletrônicas para enviar uma mensagem não solicitada (*spam*), publicidade, bem como enviar mensagens repetidamente no mesmo site. Dentre esses aspectos, os que se destacam são os inúmeros erros gramaticais, a tentativa de afetar a opinião do leitor em alguns tópicos manipuladores, o conjunto limitado de palavras semelhante no texto, além de muita poluição visual, e conteúdo(s) falso(s). Essa semelhança permite a utilização de abordagens semelhantes para filtragem de *spam* e detecção de *Fake News*.

2.2 Mineração de Texto

A Mineração de Textos, também conhecida como *Knowledge Discovery in Texts* (KDT) de tradução livre, Técnicas de Conhecimento em Textos, refere-se ao processo de extração de informação útil (conhecimento) em documentos de textos não estruturados (BARION; LAGO, 2015).

Se trata de uma subárea da Descoberta de Conhecimento em Bancos de Dados ou *Knowledge Discovery in Databases* (KDD), que visa extrair alguma informação valiosa dos conjunto de dados (MARUMO, 2018). Para Morais e Ambrósio (2007), as fases do KDD são dividas em quatro principais etapas, que são apresentadas na figura 2.1.

Figura 2.1 – Diagrama da visão geral do problema em MT com descrição de suas etapas abaixo.



Fonte: O autor, adaptado de (MARUMO, 2018)

2.2.1 Etapa 1: Amostras

Nesta etapa ocorre a identificação do problema a ser resolvido para selecionar os dados, ou amostras. Esse processo varia dependendo do domínio que está sendo tratado. Na identificação de *Fake News*, as amostras podem ser recolhidas de sites de notícias mais diversos na *internet*, além de publicações nas mídias sociais.

2.2.2 Etapa 2: Formatação dos Dados (Pré-Processamento)

Para que os dados sejam processados nas etapas seguintes é necessário que estes sejam convertidas para um formato apropriado. Os dados de treino para classificação correspondem a um conjunto de instâncias chamadas de exemplos, as mesmas possuem informações chamadas de atributos e o peso associado a essas instâncias é denominado rótulo. Na mineração de texto a frequência das palavras presentes no texto podem ser consideradas como os atributos e com isso formar um vetor de características (BECKER; TUMITAN, 2013). Os atributos podem ser formados de forma binária (com o termo presente ou não no texto) utilizando um dicionário (*bag of words*), por quantidade de vezes que aparece no texto ou usando a técnica *Term Frequency - Inverse Document Frequency*

(TF-IDF) onde o valor do atributo vai ser a frequência da palavra no texto multiplicado pelo inverso da frequência no conjunto de dados inteiro.

Outra abordagem que pode ser utilizada é a Word2Vec, mas vale ressaltar que além desta, existem outras. Ela utiliza um método implementado de *Word Embedding* com auxílio de Redes Neurais Artificiais. O *Word Embedding*, por sua vez, é um tipo de representação de palavras que permite que palavras de significados parecidos tenham representações similares. Essas representações são vetores densos de altas dimensões, onde cada entrada é uma medida entre a associação entre a palavra e o contexto onde ela se encontra (GOLDBERG; LEVY, 2014).

A Word2Vec possui dois modelos para realizar o aprendizado das *Word Embedding*: *Continuous Bag-of-Words* e *Skip-Gram*. O aprendizado do modelo *Continuous Bag-of-Words* é realizado predizendo uma palavra baseada no contexto de uma frase, enquanto o modelo *Skip-Gram* aprende prevendo um contexto baseado em uma palavra escolhida. Em ambos modelos, o aprendizado é feito utilizando *back-propagation* (MARUMO, 2018), que é explicado posteriormente na Subseção 2.3.4.1.

2.2.3 Etapa 3: Aplicação dos Algoritmos (Mineração de dados)

A mineração de dados é a fase onde são escolhidos os algoritmos para realização da mineração. Essa abordagem utiliza das técnicas de aprendizado de máquina para resolver problemas de classificação. A Seção 2.3 define melhor o conceito de aprendizado de máquina.

A metodologia dessa abordagem é feita da seguinte forma: primeiro obtêm-se a base de textos para treinamento, essa pode vir previamente classificada (aprendizado supervisionado) ou não (aprendizado não-supervisionado). Depois cada documento é representado por um vetor de características (*features*), para em seguida o algoritmo ser treinado para distinguir as características relevantes. Finalmente o algoritmo é utilizado para realizar novas previsões em novos dados (TSYTSARAU; PALPANAS, 2012).

2.2.4 Etapa 4: Avaliação dos Resultados (Pós-Processamento)

A fase final da mineração de texto é o pós-processamento, onde será avaliado o resultado obtido. Na aplicação de algoritmos de Aprendizado de Máquina (AM) em problemas reais, em geral, o conjunto de dados coletado é utilizando tanto para treino quanto para teste do modelo preditivo. Dessa forma, faz-se necessário o uso de técnicas de avaliação de classificadores (FACELI et al., 2011a). A acurácia é uma das métricas mais utilizadas para verificação de desempenho, ela mede a taxa de classificações corretas para o conjunto de teste utilizado no classificador.

Ainda de acordo com (FACELI et al., 2011a) a avaliação de medidas de desempenho

em duas classes, pode ser, verdadeiro-positivo (VP), onde o número de documentos que foram classificados como positivos realmente são da classe de positivos; verdadeiro-negativo (VN), onde o número de documentos que foram classificados como negativos, realmente são da classe de negativos; falso-positivo (FP) onde o número de documentos que foram classificados como positivos, são da classe de negativos; e por fim o falso-negativo (FN), onde o número de documentos que foram classificados como negativos, são da classe dos positivos.

Com essas definições pode-se perceber que os verdadeiros-positivos e os verdadeiro-negativos são realmente as classificações feitas corretamente, enquanto que os falso-negativos e os falso-positivos são classificações feitas erroneamente (SANTOS, 2014a).

Esse tipo de classificação remete ao uso de uma matriz de confusão. Uma matriz de confusão representa o valor real dos dados pela predição do classificador, ou seja, as linhas representam as classes verdadeiras e as colunas representam as classes preditas pelo classificador. A Tabela 2.1 mostra uma matriz de confusão para um problema de duas classes.

Tabela 2.1 – Matriz de confusão para duas classes.

		Classe Predita	
		+	-
Classe Verdadeira	+	**VP**	**FN**
	-	**FP**	**VN**

A partir dessa matriz, uma série de outras medidas podem ser usadas para medir o desempenho do classificador (FACELI et al., 2011a). Entre elas temos:

- **Taxa de acerto ou acurácia total**: Proporção de exemplos classificados corretamente do total, dado pela soma dos valores da diagonal principal da matriz.

$$ac(\hat{f}) = \frac{VP + VN}{n} \tag{2.1}$$

- **Precisão**: Proporção de exemplos positivos classificados corretamente entre todos aqueles marcados como positivos pelo classificador.

$$prec(\hat{f}) = \frac{VP}{VP + FP} \tag{2.2}$$

- ***Recall* (ou sensibilidade)**: Proporção de acerto na classe positiva, também chamado de taxa de verdadeiros-positivos (TVP).

$$recall(\hat{f}) = TVP(\hat{f}) = \frac{VP}{VP + FN} \tag{2.3}$$

- **Especificidade**: Corresponde a taxa de acerto na classe negativa, também chamado de taxa de verdadeiros-negativos (TVN).

$$esp(\hat{f}) = TVN(\hat{f}) = \frac{VN}{VN + FP} \tag{2.4}$$

A precisão e o *recall* são valores importantes e complementares para medir o desempenho de um classificador. A precisão consegue avaliar a capacidade de predição para exemplos previstos como positivos, mas não consegue avaliar os dados que também eram da classe positiva mas que não foram classificados corretamente. Enquanto que o *recall* nos diz a proporção de exemplos que eram da classe positiva e foram previstos como positivos, porém não nos diz nada quanto aos exemplos que foram incorretamente atribuídos a classe positiva (FACELI et al., 2011a).

Nas subseções que seguem serão abordados alguns algoritmos de fragmentação, utilizados neste trabalho, como é o caso do *Cross-Validation* e o *Hold-Out*.

2.2.4.1 *Hold-Out*

Também chamado de *Percent split* em (WITTEN et al., 2016), o *Hold-Out*, recebe como entrada, a informação referente à uma porcentagem. Esse valor será responsável pela divisão dos dados em subconjuntos.

Esses subconjuntos são os dados de treinamento e os dados de teste. O conjunto de treinamento será utilizado para treinar o classificador, enquanto o conjunto de teste será utilizado para validar a classificação. O *Hold-Out* é finalizado quando são obtidos os valores das predições no conjunto de teste, gerando dados estatísticos (PIMENTA et al., 2009).

2.2.4.2 *K-Fold Cross Validation*

Em problemas de classificação onde o mesmo conjunto de dados que é usado para treinar também é usado para teste, é necessário um cuidado especial, pois não é honesto classificar dados que o algoritmo de AM já tenha visto na etapa de treino. Um método alternativo ao *Hold-Out* para obter o desempenho de um classificador é a validação cruzada *k-fold cross-validation*. Nela divide-se o conjunto de dados em k partições iguais, então uma partição é usada para teste e o restante para treino. Esse processo é repetido para todas as partições afim de obter o desempenho do classificador em cada uma, para então ter uma média do desempenho final (FACELI et al., 2011a).

Um método mais sofisticado dessa técnica é o *k-fold cross validation* estratificado, que mantém cada partição com a mesma proporção de exemplos de cada classe que o conjunto de dados inteiro possui (FACELI et al., 2011a).

Existe ainda um caso extremo denominado *leave-one-out*, onde o número de partições é o mesmo do número de exemplos contidos, ou seja, o algoritmo sempre treina "deixando um exemplo de fora" e testa usando esse mesmo exemplo. Esse método é mais fiel ao desempenho final, porém demanda muito recurso computacional, principalmente se o conjunto de dados possuir muitos exemplos (FACELI et al., 2011a).

2.3 Aprendizado de Máquina

O Aprendizado de Máquina (AM) é uma subárea da inteligência artificial cujo objetivo é desenvolver técnicas computacionais sobre o aprendizado e a construção de sistemas que adquiram conhecimento de forma automática (MONARD; BARANAUSKAS, 2003). Um sistema de aprendizado é um programa que toma decisões baseado na sua experiência. Existem várias definições para aprendizado de máquina, Mitchell (1997) a define da como a capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência.

Algoritmos de aprendizado de máquina são usados em diversas tarefas que podem ser divididas em tarefas preditivas e tarefas descritivas. As tarefas preditivas, que são denominadas de aprendizado supervisionado, são usadas para predizer o valor de novos exemplos a partir de dados já rotulados. Para isso, é necessário um agente especialista para avaliar a capacidade de previsão do algoritmo. Nas tarefas descritivas, denominadas de aprendizado não supervisionado, o objetivo é explorar e descrever certo conjunto de dados, não sendo necessário um atributo de saída (FACELI et al., 2011a).

Nas subseções que seguem serão apresentados alguns classificadores supervisionados existentes na literatura e que serão utilizados neste trabalho.

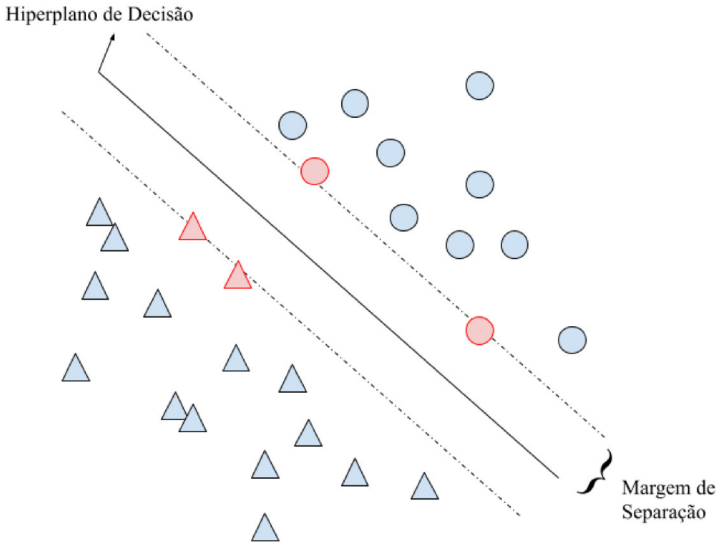
2.3.1 SVM

As Máquinas de Vetores de Suporte (SVMs), do Inglês *Support Vector Machines* constituem uma técnica de aprendizado supervisionado que utiliza resultados fornecidos pela Teoria do Aprendizado Estatístico (TAE) para classificação, desenvolvida por (VAPNIK, 2013) a partir de estudos iniciados no trabalho de (VAPNIK; CHERVONENKIS, 2015) e fundamentada nos princípios da Minimização do Risco Estrutural (SRM), do Inglês *Structural Risk Minimization*. Essa teoria estabelece uma série de princípios que devem ser seguidos na obtenção de classificadores com boa generalização, desta meneira um classificador com boa generalização terá a capacidade de prever corretamente a qual classe pertence os novos dados, levando em consideração o mesmo domínio em que o aprendizado ocorreu (LORENA; CARVALHO, 2007).

Dado um conjunto de dados de treinamento, previamente demarcados em relação as duas classes existentes, o SVM prediz a classe que o atual conjunto de dados pertence,

construindo um modelo que atribui novos exemplos a uma categoria ou outra. A representação gráfica desse algoritmo é apresentada como pontos no espaço. Esses pontos são separados pelo chamado hiperplano de decisão, que deve ser claramente amplo, tanto quanto possível. Desta forma, cada categoria estará de um lado desta separação. A Figura 2.2 apresenta a demonstração gráfica precedentemente descrita.

Figura 2.2 – SVM linear com fronteira de decisão maximizada.



Fonte: O autor.

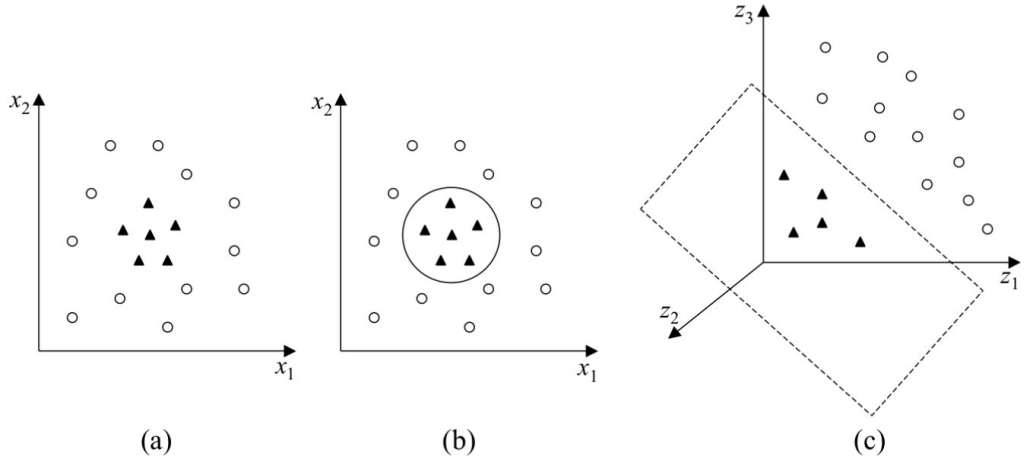
Para melhor entendimento suponha que o conjunto de dados, desta mesma Figura (2.2), possua poucos atributos e somente duas classes possíveis: Δ e O. Nesse exemplo, o conjunto irá formar um plano bidimensional onde o SVM irá buscar uma reta (de acordo com Santos (2014b) para problemas com dimensões maiores, essa fronteira de decisão é chamada de hiperplano) que separe as duas classes, essa reta deverá ter a maior distância possível de todos os pontos do conjunto. A mesma Figura mostra um exemplo de conjunto linearmente separável, onde a reta busca a maior distância entre os pontos. Os vetores onde se encontram os atributos em vermelhos são chamados de vetores de suporte *(Support Vectors)*. Vale ressaltar que quanto mais ajustada for superfície de decisão aos dados do conjunto de treinamento, isto é, quanto mais complexo for o hiperplano de decisão, dessas funções no espaço de entrada dos dados, maior será o risco estrutural (CHERKASSKY; MULIER, 1999).

Para problemas que possuem ruídos, é necessário o uso de uma margem suave para obter o hiperplano. Para isso utiliza-se uma folga de erro ξ que permite ruídos fora da zona de sua classe, na Figura 2.2 essa margem é descrita como "margem de separação". A folga é regularizada com uma constante C que impõe um peso para o erro permitido. Os valores da folga e do erro são inversamente proporcionais, ou seja, quanto maior for o valor

de C, menor será o ξ permitido. Essa configuração eleva ainda mais o ajuste da fronteira de decisão. De acordo com Faceli et al. (2011b) valor padrão para a constante C é 1 .

Na maioria dos problemas de classificação, os dados não são linearmente separáveis, sendo necessário uma abordagem diferente da abordagem do SVM linear. Para problemas onde não é possível achar um hiperplano separador, utiliza-se as SVM's não-lineares, onde é utilizada uma função ϕ que mapeia os dados e faz uma transformação não-linear para uma dimensão maior. Esse novo espaço é chamado de espaço de características *(feature space)*. Uma escolha apropriada da função ϕ faz com que o espaço de característica seja linearmente separável e possa ser utilizado uma SVM linear (FACELI et al., 2011b). A Figura 2.3 apresenta um conjunto de dados não-linear, sua fronteira para duas dimensões e seu espaço de características sendo separado por um hiperplano.

Figura 2.3 – (a) Conjunto de dados não linear; (b) Fronteira não-linear; (c) Fronteira linear no espaço de características.



Fonte: (CHERKASSKY; MULIER, 1999).

Ainda de acordo com Faceli et al. (2011b) a função ϕ é também chamada de *kernel*. Os *kernels* mais utilizados na literatura são os polinomiais, os de função base radial *Radial Basis Function* (RBF) e os sigmoidais. Existe também o *kernel* linear, mas o mesmo normalmente é associado como um caso especial do RBF. O SVM possui vantagens sobre outros classificadores por sua facilidade de generalização com dados de grande dimensão, sobre os quais as outras técnicas obtêm classificadores ineficientes. No entanto, as desvantagens principais do SVM são a sensibilidade na escolha dos parâmetros e a dificuldade de interpretação do modelo gerado pela técnica.

2.3.2 *Naive Bayes*

Para iniciar o aprendizado *Naive Bayes* antes se faz necessário a aplicação do Teorema de Bayes para prever a probabilidade de dado exemplo pertencer a uma classe.

As notações que seguem foram apresentadas no trabalho de Mitchell (1997), estas são utilizadas para definição do Teorema de Bayes:

- Dada uma hipótese h e um conjunto de treino D tem-se que $P(h)$ denota a probabilidade inicial da hipótese h antes de observarmos o conjunto de dados D.
- Similarmente, $P(D)$ denota a probabilidade do conjunto de treino D ser observado.
- $P(D|h)$ denotará a probabilidade de observamos o conjunto D dado que a hipótese h ocorre, ou seja, a chance de ocorrer D dado que h ocorre.
- Finalmente, $P(h|D)$ é a probabilidade do evento h ocorrer, dado o conjunto de treino D. O teorema de Bayes provê uma maneira de calcular $P(h|D)$ sabendo a priori a probabilidade $P(h)$, junto com $P(D)$ e $P(D|h)$.

Teorema de Bayes:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{2.5}$$

Como o Teorema de Bayes fornece uma maneira de calcular a probabilidade posterior de uma hipótese dado um conjunto de treinamento, o classificador pode prever os rótulos dos dados buscando aqueles dados que possuem maior probabilidade de ocorrer. Em um problema de classificação devemos encontrar a classe y_i que maximize a probabilidade dado um conjunto $\mathbf{x}$ (FACELI et al., 2011a). Formalmente obtemos:

$$y = arg\,max\ P(y_i|\mathbf{x}) \tag{2.6}$$

Aplicando ao Teorema de Bayes obtemos:

$$y = arg\,max\ \frac{P(\mathbf{x}|y_i)P(y_i)}{P(\mathbf{x})} \tag{2.7}$$

O denominador, $P(\mathbf{x})$, pode ser removido, pois será sempre o mesmo para todas as classes, não afetando os valores das probabilidades de cada classe.

$$y = arg\,max\ P(\mathbf{x}|y_i)P(y_i) \tag{2.8}$$

O classificador obtido usando esta função discriminante é chamado de classificador *Naive Bayes*. O termo *naive* (ingênuo) vem da hipótese de que os atributos de um exemplo são independentes entre si, o que normalmente não ocorre em problemas de classificação (FACELI et al., 2011a).

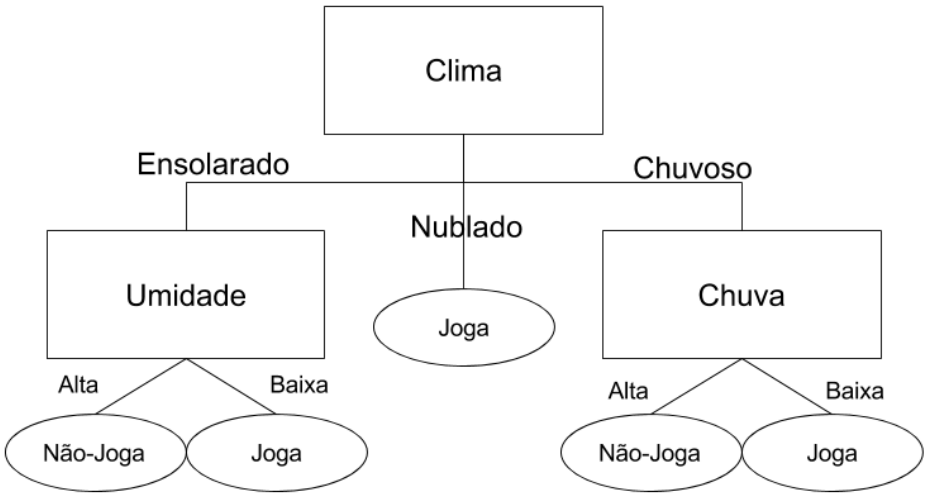
2.3.3 Árvore de Decisão

Árvore de decisão é uma técnica de aprendizado que consiste na abordagem dividir para conquistar para resolver um problema. Um problema complexo é dividido em subproblemas mais simples, os quais são divididos novamente usando a mesma estratégia. As soluções dos subproblemas podem ser combinadas, formando uma árvore, para termos a solução do problema complexo (FACELI et al., 2011a).

Em tarefas preditivas, os nós da árvore de decisão envolvem o teste de um atributo em particular. Nós folhas são os responsáveis por dar a classificação que se aplica a todas as instâncias que levam a folha. Para classificar uma instância desconhecida, é percorrida a árvore de acordo com os valores dos atributos testados em seus nós sucessivos, e quando um nó folha é alcançado, a classificação é dada de acordo com a classe assinada no nó folha (WITTEN; FRANK, 2005).

Um exemplo simples de uma árvore de decisão pode ser visto na Figura 2.4 onde o problema se define em resolver se o clima está favorável para uma partida de futebol. Uma árvore de decisão abrange todo o estado de instâncias, ou seja, para qualquer valor de entrada sempre haverá uma predição do rótulo do exemplo (FACELI et al., 2011a).

Figura 2.4 – Exemplo de árvore de decisão.



Fonte: O autor.

A construção de uma árvore de decisão é um problema recursivo. Primeiro escolhemos um atributo para ser a raiz da árvore e criamos um galho para cada possível valor. Para cada galho o processo é repetido recursivamente. Se em algum instante todas as instâncias de um nó possuem a mesma classificação, o desenvolvimento da árvore naquele nó é parado (WITTEN; FRANK, 2005).

A decisão de como escolher os atributos para cada nó é guiada por uma medida

de "*goodness of split*", que indica quão bem um atributo separa as classes (FACELI et al., 2011a). Existem várias técnicas de divisão de atributos, dentre estas será destacada a técnica da divisão baseada no *Ganho de Informação*.

O conceito principal dessa técnica remete ao conceito de entropia. A entropia descreve o grau de desordem ou aleatoriedade em um sistema. Em um conjunto de treinamento, o objetivo é encontrar o atributo com menor entropia, ou seja, aquele que tenha menos aleatoriedade e divida melhor o problema (FACELI et al., 2011a). A entropia de um valor a_i de um atributo A é dado por:

$$H(y|A = a_i) = -\sum_i p_i \times log_2(p_i) \tag{2.9}$$

onde p_i é a proporção de cada classe y para o valor a_i do atributo A. Para calcular a soma ponderada da entropia do atributo A usamos:

$$H(A) = \sum_i q_i \times H(y|A = a_i) \tag{2.10}$$

onde q_i é a proporção de cada classe y no atributo a. O ganho de informação é obtido pela diferença da entropia do conjunto de exemplos pela soma ponderada da entropia dos atributos. Com essa heurística de decisão, o nó com maior ganho de informação será o nó raiz e, recursivamente, esse nó será dividido novamente pelo atributo com maior ganho de informação. Ao final dessa técnica, obtém-se a árvore de decisão referente a base de treino utilizada.

2.3.4 Redes Neurais Artificiais

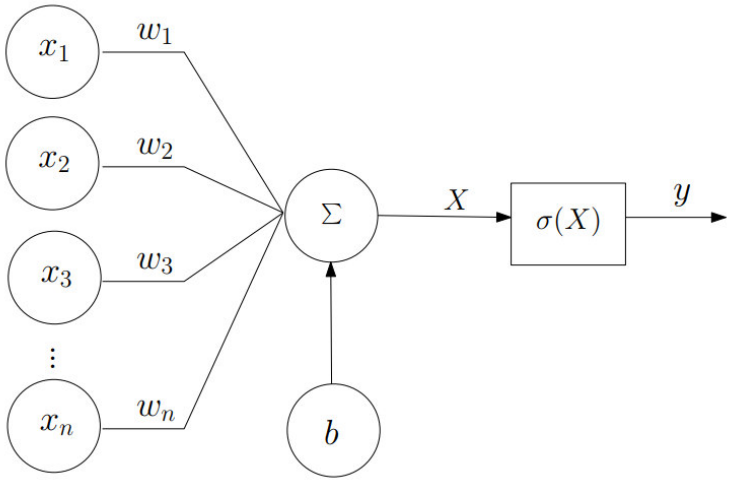
De acordo com Haykin (2007), uma rede neural é um computador maciçamente paralelamente distribuído, constituído de unidades de processamento simples. O mesmo se assemelha ao cérebro pois seu conhecimento é adquirido pelo ambiente através de um processo e aprendizagem e as forças de conexões entre neurônios são utilizadas para armazenar o conhecimento adquirido, essas forças são chamadas de pesos sinápticos.

Outra definição para redes neurais artificiais é dada por Gurney (1997), definindo como uma montagem interconectada de nós, cujo sua funcionalidade é baseada em redes neurais biológicas. O processamento da rede funciona a partir da interconexão desses nós através de pesos que são obtidos e adaptados através de padrões de treinamento, esse processo é denominado aprendizado.

As redes neurais artificiais foram desenvolvidas como uma generalização a partir de modelos matemáticos de sistemas nervosos biológicos. Seus nós são chamados de neurônios, e a modelagem matemática dos seus elementos são as sinapses que são representadas como os pesos nas conexões dos nós que modulam o sinal de entrada do neurônio e a característica não linear dos neurônios biológicos é representada por uma função de transferência (ABRAHAM, 2005).

A Figura 2.5 demonstra a arquitetura básica de uma rede neural artificial. Nela é possível notar que o neurônio possui vários elementos que se iniciam pelos sinais de entrada, que são valores numéricos, dos quais são transferidos pelos pesos sinápticos de outros neurônios ou de um sinal inicial. Esses valores são ponderados pelos pesos para serem somados entre si e um valor denominado *bias*. O valor final é aplicado a uma função de ativação (ou função de transferência) gerando uma nova saída. A saída deste neurônio pode ser a entrada de um novo neurônio ou a saída final do modelo (SOUSA, 2018).

Figura 2.5 – Arquitetura de uma rede neural artificial.



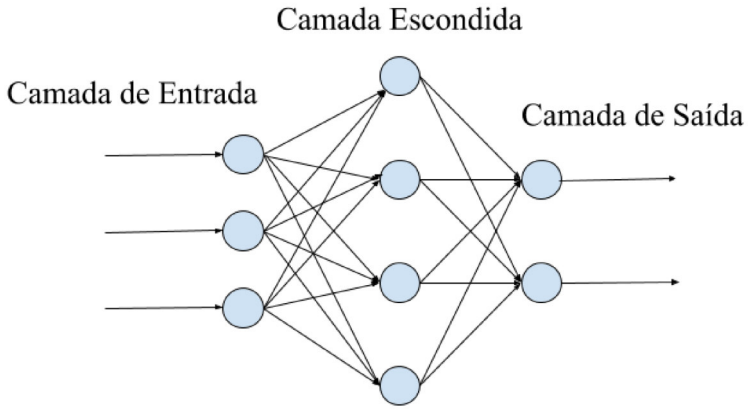
Fonte: (KOVALESKI, 2018).

A partir da Figura 2.5 é possível modelar uma equação matemática do funcionamento de um neurônio artificial, a Equação 2.11 demonstra a modelagem desta equação onde os valor x_i, y, w_i, b e φ são os sinais de entrada, a saída do neurônio, o peso correspondente a entrada, o valor *bias* adicionado e a função de ativação no neurônio, respectivamente.

$$y = \varphi(\sum_{i=1}^{n} x_i w_i + b) \tag{2.11}$$

Um neurônio resolve apenas problemas linearmente separáveis, para isso é necessário associar dezenas de neurônios em várias camadas interconectados para sanar o problema (ABRAHAM, 2005). Esses modelos são denominados de *Multilayer Perceptron* (MLP), a Figura 2.6 retrata um exemplo de um MLP que possui três camadas de três tipos, a camada de entrada em que os sinais codificados são inseridos, a camada escondida onde os dados são processados e transferidos para a camada seguinte que é a camada de saída, a qual resulta na classificação final para o sinal de entrada (SOUSA, 2018).

Figura 2.6 – *Multilayer Perceptron.*



Fonte: O autor, adaptado de (ABRAHAM, 2005).

Para utilização de redes neurais em aprendizagem de máquina, é necessário fazer atualizações nos pesos sinápticos, pois os mesmos são os responsáveis por ditar a saída da rede para determinado problema. O principal método para treinamento e, portanto, a atualização dos pesos é o algoritmo *Back-Propagation*.

2.3.4.1 *Back-Propagation* e Método do Gradiente Descendente

De acordo com MARUMO (2018) o *back-propagation* é um algoritmo supervisionado de treinamento. Em sua primeira fase, para frente (*forward*), são gerados valores de saída da rede, e na segunda fase, para trás (*backward*), são utilizados esses valores de saída e os valores desejados para calcular os erros e atualizar os pesos sinápticos. Os erros são calculados por una função de custo, onde a saída da rede é comparada com a saída real e os ajustes podem ser feitos utilizando algoritmos de otimização, como por exemplo o método do gradiente descendente(BRAGA, 2000).

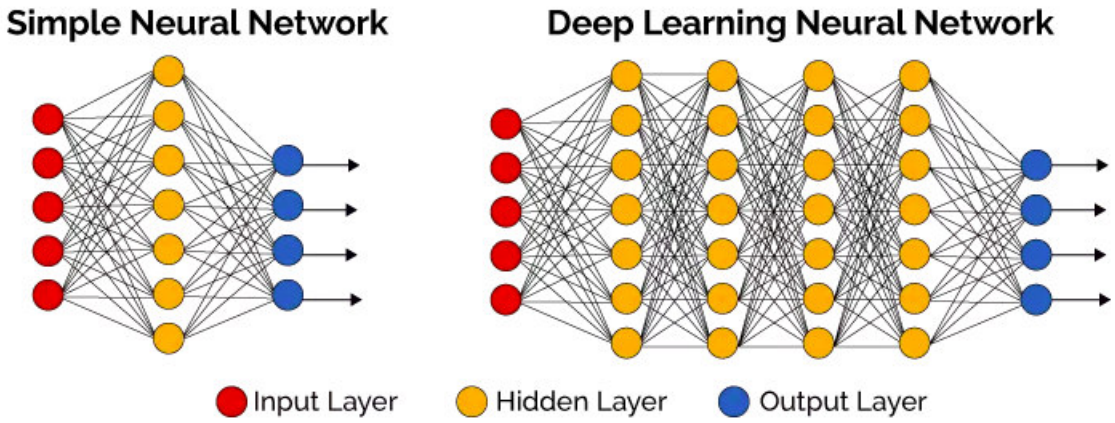
O método do gradiente descendente busca encontrar o valor de mínimo local, visto que este valor é diretamente proporcional ao valor do erro. Assim o algoritmo *back-propagation* procura minimizar o erro obtido pela rede ajustando pesos e limiares para que eles correspondam às coordenadas dos pontos mais baixos da superfície de erro, essa direção é encontrada calculando a derivada do função de custo indicando a direção em que a mesma decresce (BRAGA, 2000). A função de custo determinam o peso dos erros, é ela que escolhe a forma como penalizar os resultados que foram preditos de maneira diferente dos valores reais. MARUMO (2018) afirma que quanto maior o valor de erro, menor é o aprendizado do modelo.

2.4 Deep Learning

Deep Learning (DL) ou Aprendizagem Profunda é a denominação para redes neurais mais robustas, ou seja, com muitas camadas intermediárias. Se baseia em aprender diferentes níveis de abstração, utilizando várias camadas de processamentos não lineares (DENG; YU, 2014).

Com isso, DL trata-se de um aprendizado de redes neurais com arquiteturas mais profundas conseguindo extrair melhor as características de um conjunto de dados. A Figura 2.7 apresenta uma comparação de uma rede neural simples com uma DL.

Figura 2.7 – Rede neural simples e DL.



Fonte: (Data Science Academy, 2019)

De acordo com Deng e Yu (2014) existem três principais razões para a popularidade da DL atualmente:

1. O aumento drástico do poder computacional por unidades de processamento gráficos (GPU).
2. O aumento significativo do tamanho dos dados utilizados para treinamento
3. Os recentes avanços em aprendizado de máquina e processamento de sinais e informações que facilitaram a construção de redes mais complexas.

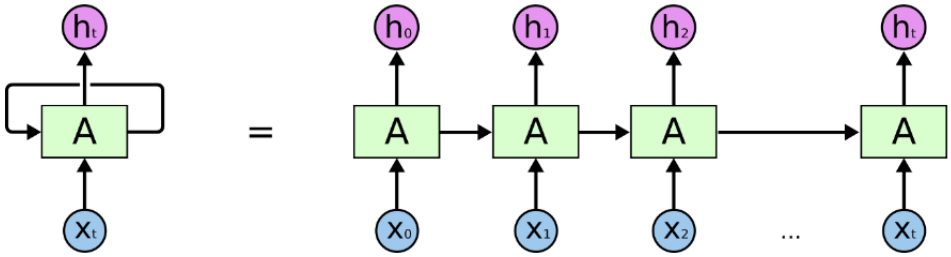
A seção a seguir apresenta uma descrição de um dos algoritmos de DL utilizados na literatura.

2.4.1 LSTM

Antes de falar sobre LSTM, é necessário introduzir sobre Redes Neurais Recorrentes (RNN - *Recurrent Neural Networks*), que são um tipo de rede que possui *loops* que permitem

que a informação persista. Sua arquitetura está descrita na Figura 2.8 onde a rede recebe as entradas x_t, que produzem as saída h_t que serão utilizadas para alimentar as entradas na próxima iteração. Esse *loop* gera uma memória sobre os resultados passados criando uma dependência na sequência dos dados (MARUMO, 2018). Com isso, essas redes são mais utilizadas em tarefas de processamento de linguagem natural, pelo fato de modelarem sequências em unidade (MENDES, 2018).

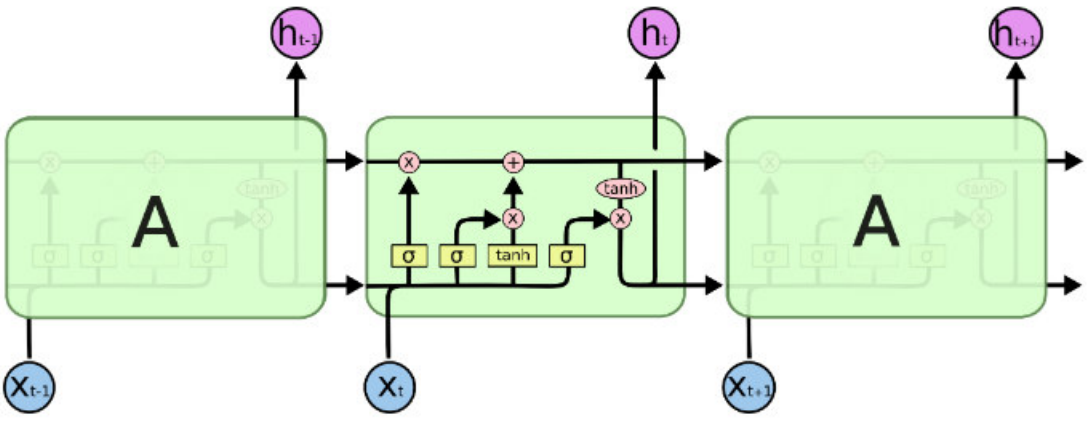
Figura 2.8 – Arquitetura de uma RNN.



Fonte: (OLAH, C., 2015)

Embora seja recorrente, a RNN também possui limitações. As vezes só é necessária a informação anterior para a realizar a predição mas há casos em que é necessário mais contexto, ou seja, precisa de uma memória que persista por toda a sequência. As redes *Long Short Term Memory* (LSTM) surgiram para aprender dependências de longo prazo através de uma célula extra de memória que transmitirá para os próximos estágios (OLAH, C., 2015). A Figura 2.9 apresenta a arquitetura de uma rede LSTM onde a linha horizontal no topo é a célula de memória.

Figura 2.9 – Arquitetura de uma LSTM.



Fonte: (OLAH, C., 2015)

A LSTM possui a habilidade de adicionar e remover informações da célula de memória através de *gates*. A LSTM pode ser dividida em três estágios:

- *Forget Gate*: Esse primeiro estágio é o que decide o quanto de informação será jogada fora da célula de memória. É controlado por uma função sigmoide que serve como um peso para decidir o quando daquela informação permanecerá (OLAH, C., 2015).
- *Input Gate*: Neste estágio, os valores da célula de memória serão atualizados através da multiplicação de duas funções, uma sigmóide e uma tangente hiperbólica. Elas são multiplicadas entre si e somadas a célula de memória (OLAH, C., 2015).
- *Output Gate*: Estágio final da LSTM onde serão decididas a saída daquela camada. Ela é dada pelas entradas da rede passando por uma função sigmóide multiplicado pela célula de memória passando por uma tangente hiperbólica (OLAH, C., 2015).

Com essa capacidade de guardar informação por mais tempo, elas são essenciais em grandes textos por facilitar a associação de padrões nos mesmos (MARUMO, 2018).

3 Metodologia

Este trabalho tem como objetivo a detecção de *fake news* em notícias utilizando técnicas de inteligência artificial. Utiliza-se a abordagem de aprendizado profundo, além de fazer uma comparação com os algoritmos tradicionais de aprendizado de máquina.

3.1 Obtenção do Conjunto de Dados

Para realizar o treinamento das redes neurais é necessário uma base da dados com notícias reais e notícias falsas. Neste trabalho foi utilizado o "Fake.Br Corpus", que é um conjunto de dados de notícias verdadeiras e falsas criado por Monteiro et al. (2018). Este conjunto de dados possui 3600 notícias reais e 3600 notícias falsas. Essas notícias estão alinhadas por assunto e por tamanho para evitar o viés sobre os dados. Todos os detalhes da obtenção do conjuntos de dados pode ser visto no trabalho dos autores. Desta forma, o conjunto de dados ficou dividido como mostra a Tabela 3.1

Tabela 3.1 – Descrição do Fake.Br Corpus

Categoria	Quantidade	Porcentagem
Política	4180	58,0
TV e Celebridades	1544	21,4
Sociedade e Notícias diárias	1276	17,7
Ciência e Tecnologia	112	1,5
Economia	44	0,7
Religião	44	0,7
Total	7200	100

Realizou-se neste trabalho algumas alterações no conjunto de dados pois notou-se que havia, embora bem poucas, algumas repetições. As mesmas foram removidas e algumas alteradas para manter o alinhamento do conjunto de dados. Assim havia 7200 notícias divididas igualmente entre verdadeiras e falsas onde cada par de notícia foi truncado pelo tamanho do menor, procedimeno necessário pois de acordo com Monteiro et al. (2018) as notícias reais tendem a serem bem maiores comparadas com as Fake news.

3.2 Pré-Processamento

A fase de pré-processamento realizada nas notícias foi feita utilizando a linguagem de programação Python 3.6 através da biblioteca de tratamento de texto NLTK (*Natural Language Toolkit*) (BIRD; KLEIN; LOPER, 2009) que oferece uma variedade de técnicas

de processamento de linguagem natural. Desta forma, o pré-processamento se deu nos seguintes passos:

1. **Padronização dos termos**: Nessa etapa é onde as palavras de todas as notícias são padronizadas através da eliminação de pontuação, eliminação de acentos, caracteres especiais e uniformização das letras, passando todas para caixa baixa.

2. **Remoção de *Stop Words***: Nessa etapa são removidas as *stopwords*, que são palavras consideradas irrelevantes ao texto, pois não tem sentido próprio quando consideradas sozinhas. Como exemplos temos os artigos e as preposições.

3. ***Stemming***: *Stemming* é o processo de reduzir palavras flexionadas ao seu radical (*stem*), dessa forma as palavras flexionadas (como plural, sufixo temporal, formas de gerúndio) são reduzidas ao seu radical.

Com este processo, as notícias foram organizadas em forma de listas, onde cada lista era uma notícia e cada elemento da lista era uma palavra reduzida ao seu radical (que não foi eliminada nas etapas 1 e 2). A Tabela 3.2 demonstra como ficou parte de uma notícia após a esses passos citados.

Tabela 3.2 – Exemplo de Fake News pré-processada

Notícia	Pré-Processamento
Temer resolve o problema de Luislinda: "liberdade, ainda que tardia". A ministra dos Direitos Humanos Luislinda Valois entrou para a galeria dos "sem noção" ao pedir para acumular o salário da função de ministra com a aposentadoria de desembargadora, o que daria R$ 61 mil, muito além do teto constitucional.	['tem', 'resolv', 'problem', 'luislind', 'liberdad', 'aind', 'tard', 'ministr', 'direit', 'human', 'luislind', 'valoil', 'entr', 'gal', 'noc', 'ped', 'acumul', 'salari', 'func', 'ministr', 'aposentad', 'desembarg', 'dar', '61', 'mil', 'alem', 'tet', 'constituc']

Nota-se na Tabela 3.2 que nomes próprios acabam sofrendo nesta etapa de pré-processamento. É possível verificar que o nome "*Temer*" foi reduzido para "*tem*" devido a escrita ser igual ao do verbo "*temer*". Esse trabalhou não visou resolver nomes próprios.

3.3 Vetores de Características

Para criação dos vetores de características das notícias utilizou-se duas abordagens, a primeira para utilização dos métodos tradicionais de aprendizado de máquina e a segunda para o treinamento da DL. As subseções a seguir definiram os passos de cada abordagem.

3.3.1 Abordagem A

Na Abordagem A, criou-se primeiramente uma *bag of words* com todas as palavras que ocorreram no conjunto de treinamento.

A *Bag Of Words* deste trabalho contém palavras que aparecem no conjunto de textos com frequência mínima de três vezes. Se uma palavra aparece apenas duas vezes em todo o corpus de treinamento, ela é desconsiderada. Com isso, a dimensionalidade do vetor de características é diminuída.

O vetor de características de cada notícia é formado utilizando a técnica *Term Frequency - Inverse Document Frequency* (TF-IDF). Essa técnica consegue avaliar a importância de uma palavra contida em um texto e sua relevância em todo o conjunto de treino. Inicialmente, é calculada a frequência que um termo aparece em uma notícia. Suponha uma notícia x em que uma certa palavra p_i aparece em seu conteúdo, sua frequência será calculada a partir da Equação 3.1.

$$TF(p_i|x) = \frac{N(p_i|x)}{N(x)} \tag{3.1}$$

onde $N(p_i|x)$ é a número de vezes que a palavra p_i aparece na notícia x e $N(x)$ é a quantidade de palavras na notícia x. Para calcular o IDF suponha novamente a palavra p_i. Precisa-se saber em quantas notícias essa mesma palavra aparece e o número total de notícias no conjunto de treino. Com isso, o inverso da frequência no documento é calculado a partir da Equação 3.2.

$$IDF(p_i) = \frac{T}{T(p_i)} \tag{3.2}$$

onde T é a quantidade total de notícias no conjunto de treino e $T(p_i)$ é a quantidade total de notícias que aparece a palavra p_i. Por fim, o cálculo do TF-IDF se dá pela Equação 3.3.

$$TF - IDF(x|p_i) = TF(x|p_i) \times \log_{10}(IDF(p_i)) \tag{3.3}$$

Com esse método, um exemplo de como ficaria a matriz de atributos pode ser vista na Tabela 3.3 onde os dados da matriz foram normalizados.

Tabela 3.3 – Exemplo da matriz de atributos da Abordagem A.

'expuls'	**'kat'**	**'abr'**	**'pmdb'**	**'legend'**	...	**Classificação**
1	1	0,815625	0,76111111	0,525	...	Fake
0,38095238	0,15873016	0,51785714	0,36243386	1	...	Real
0,09756098	0	0	0	0	...	Fake
0	0	0	0	0	...	Real

Desta forma, cada notícia corresponderá a uma linha na matriz de atributos e as palavras na *bag of words* corresponderam as colunas. Assim a dimensão da matriz será dada pelo número de notícias pelo número de palavras na *bag of words*.

3.3.2 Abordagem B

A Abordagem B também necessita de uma *bag of words* com todas as ocorrências das palavras no conjunto de treinamento. Como na Abordagem A, esta também contém as palavras que aparecem no conjunto de dados com frequência mínima de três vezes.

O vetor de característica nessa abordagem será diferente, pois utiliza-se a técnica de *Word Embedding*, com isso, o vetor de característica de cada notícia será um conjunto de inteiros representando o índice de cada palavra da notícia na *bag of words*. A Tabela 3.4 apresenta exemplos da representação das notícias desta forma.

Tabela 3.4 – Exemplos do vetor de característica da Abordagem B

Notícia	**Vetor de Característica**	**Classificação**
1	$[2, 3, 12, 28, 206, 1, \cdots]$	Fake
2	$[19, 268, 1, 138, 64, 524, \cdots]$	Real
3	$[0, 731, 1120, 38, 9, 10, \cdots]$	Fake
4	$[98, 693, 43, 50, 65, 97, \cdots]$	Real

É possível notar na Tabela 3.4 que com essa abordagem a ordem das palavras nas notícias é preservada, pois há apenas a substituição de cada uma pelo seu índice na *bag of words*, esse tipo de informação foi perdida utilizando a Abordagem A. Para que os vetores de característica possuam o mesmo tamanho, é definido um tamanho máximo para eles. Assim aqueles que não tiverem o tamanho máximo serão preenchidos com zeros enquanto que os que tiverem tamanho maior serão cortados para que todos tenham um tamanho fixo. O número 0 representa as palavras desconhecidas para a *bag of words*.

Para este trabalho o tamanho selecionado para o vetor de característica foi de 350, assim, a matriz de atributos terá 350 colunas.

3.4 Treinamento

Feita a montagem das matrizes de atributos inicia-se a fase de treinamento. O treinamento foi feito utilizando as bibliotecas para Python *Scikit-Learn* (PEDREGOSA et al., 2011) para os algoritmos tradicionais e *TensorFlow* (TensorFlow: library, 2019) para os algoritmos de DL, para isso foi necessário dividir os dados para treinamento e teste dos algoritmos. Como foram feitas duas abordagens, para cada uma teremos um tipo de divisão de dados, a Abordagem A será para os algoritmos clássicos de aprendizagem de máquina, enquanto que a Abordagem B será para o treinamento das redes neurais.

Como a Abordagem B utiliza redes neurais, foi necessário dividir os dados em dados de treinamento, de validação e de teste, e para a comparação das duas abordagens precisamos que ambas passem pelos mesmos dados de treinamento, com isso particionou-se os dados em em 80% para treinamento, 10% para validação e 10% para teste. Assim a divisão do conjunto de notícias da base de dados Fake.Br ficou como mostrado na Tabela 3.5.

Tabela 3.5 – Divisão do conjunto de dados Fake.Br

Treinamento	5760
Validação	720
Teste	720
Total	7200

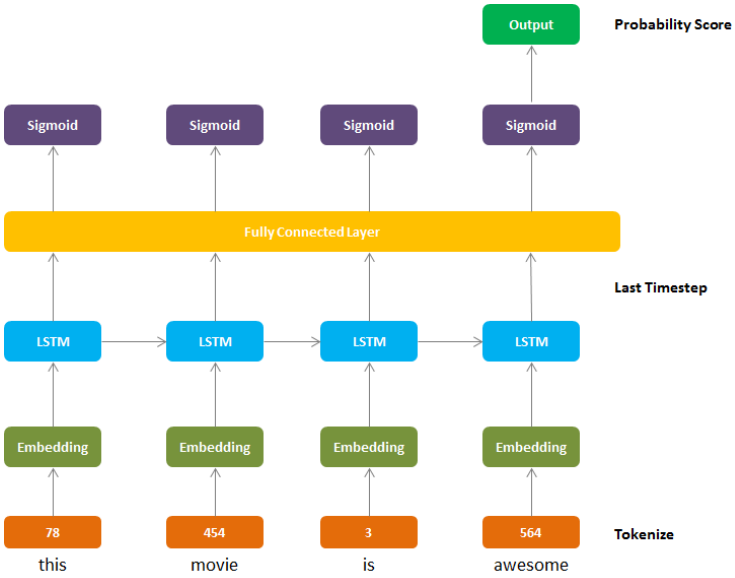
Antes de realizar a divisão, os dados foram misturados para que os conjuntos tenham notícias de todas as categorias da Tabela 3.1 e a divisão foi estratificada, ou seja, manteve-se a proporção igual de notícias reais e falsas em todos os conjuntos.

Os primeiros experimentos foram com os algoritmos tradicionais (SVM, Árvore de Decisão, *Naive Bayes*) onde foi realizado o treinamento utilizando a matriz de atributos gerados pela Abordagem A no conjunto de treinamento junto com o conjunto de validação descrito na Tabela 3.5. Após o treinamento foi realizado a predição no conjunto de teste da mesma Tabela, com elas, foram feitas as medidas de desempenho tradicionais.

Com a matriz da Abordagem B foram realizados os treinamento das redes neurais. A escolha da arquitetura é um processo árduo, então foi realizado o treinamento de 36 redes através de um *loop* e verificado o desempenho de cada uma nos dados de validação, almejando escolher as três arquiteturas com melhor desempenho.

As escolha das arquiteturas das redes se baseou em seus parâmetros, a rede precisaria passar por uma camada de *Embedding* visando representar as palavras em um vetor de dimensão previamente definida, em seguida passaria por uma quantidade N de camadas LSTM aprendendo a a co-ocorrência das palavras nas notícias, por fim a rede pode, ou não, passar por uma camada comum densa. A Figura 3.1 representa a arquitetura que essa rede genérica teria.

Figura 3.1 – Arquitetura genérica da rede neural.



Fonte: (Samarth Agrawal, Towards Data Science, 2019)

Seguindo o fluxograma, foram escolhidas 2 tipos de camadas *Embedding*, 3 opções de LSTM, 1 ou nenhuma camada densa e três opções quanto a quantidade de nós em cada camada. As opções escolhidas para as arquiteturas podem ser vistas na Tabela 3.6 onde a combinação de cada uma gerou 36 arquiteturas. Entre cada etapa da Figura 3.1 foi inserido um *dropout* de 0.2 para evitar o *overfitting* ao conjunto de treinamento.

Tabela 3.6 – Características das redes criadas.

Característica da Arquitetura	**Opções**
Dimensão do vetor *Embedding*	[64, 128]
Dimensão das camadas LSTM e densas	[32, 64, 128]
Quantidade de camadas LSTM	[1, 2, 3]
Quantidade de camadas densas	[0, 1]
Total	**36 arquiteturas**

Feito esse processo, deseja-se escolher as três melhores arquiteturas de acordo com a acurácia e custo de cada uma em suas execuções. Cada arquitetura foi treinada com o conjunto de dados por 80 épocas, que foi um tempo computacional razoável e não houve melhoras significativas na acurácia. Escolhida as três arquiteturas, será feito a predição no conjunto de teste para comparação com os outros algoritmos.

4 Resultados

O resultado inicial das execuções é dado utilizando a Abordagem *A* descrita na Seção 3.3.2 onde foram utilizados os algoritmos de classificação Árvore de Decisão, *Naive Bayes* e SVM. Foi utilizado o algoritmo *GridSearchCV* da biblioteca *scikit-learn* para obtenção do melhor valor para os parâmetros que utiliza o método *K-Fold Cross Validation* com $k = 3$ (Valor padrão do algoritmo *scikit-learn*) para cada combinação de parâmetros. A execução desse algoritmo levou 165 minutos.. A Tabela 4.1 apresenta o resultado inicial da execução para os dados de Teste.

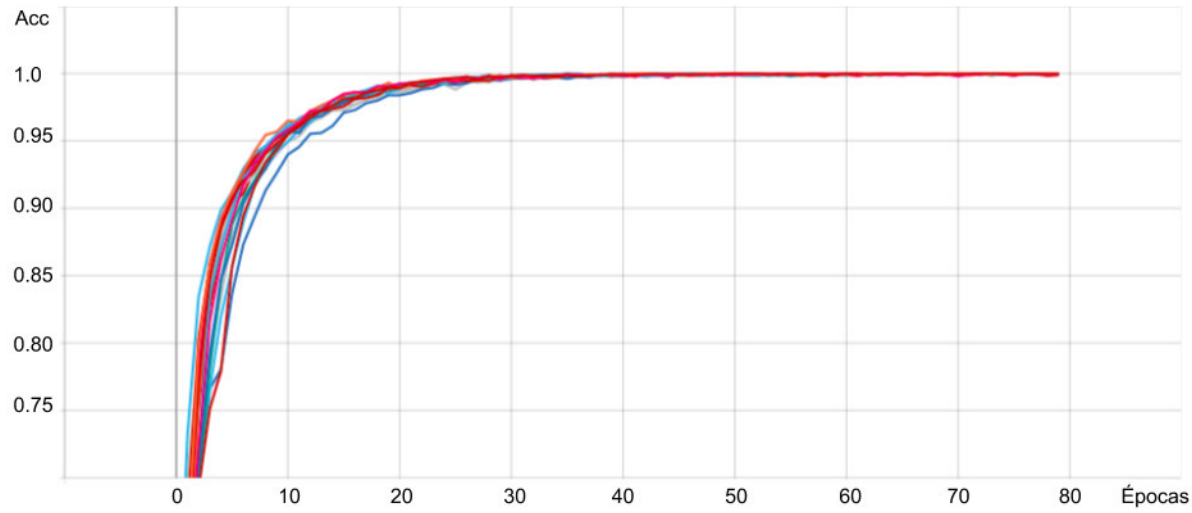
Tabela 4.1 – Resultados utilizando a Abordagem *A*.

Algoritmo	**Acurácia**	**Precisão**	**Recall**	**Especificidade**
Árvore de Decisão	77,77%	0,7793	0,775	0,7805
Naive Bayes	71,25%	0,7297	0,675	0,75
SVM	**90,13%**	**0,8753**	**0,9361**	**0,8667**

Do resultado dos algoritmos clássicos, o algoritmo que obteve melhor desempenho é o SVM que conseguiu 90% de acurácia nos dados de teste, valor acima do esperado, enquanto que os outros dois se mantiveram na faixa dos 70%, valor abaixo do esperado. É possível notar também que o SVM teve desempenho superior em todos as outras métricas, demonstrando que conseguiu definir bem um hiperplano separador dos dados. O resultado da execução do SVM confirmou os resultados obtidos por Monteiro et al. (2018) se igualando em acurácia.

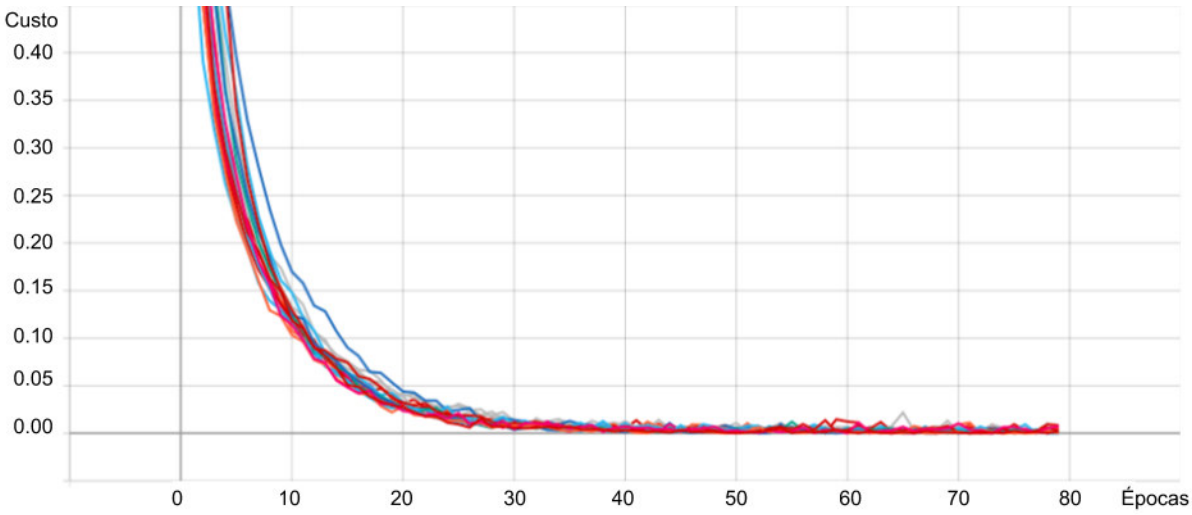
Para a obtenção dos resultados utilizando a Abordagem B foram treinadas as 36 arquiteturas de redes descritas na Tabela 3.6. As redes foram treinadas com o conjunto de treino e testadas com o conjunto de validação. Todas foram treinadas por 80 gerações, pois a partir de 80 elas não obtiveram melhoras de desempenho no conjunto de validação e o custo no mesmo continuou crescendo, o tempo de treinamento total das redes foi de 51 horas. As Figuras 4.1 e 4.2 apresentam a acurácia e o custo das redes nos dados de treinamento e as Figuras 4.3 e 4.4 apresentam a acurácia e o custo das redes nos dados de validação.

Figura 4.1 – Acurácia das redes no conjunto de Treinamento.



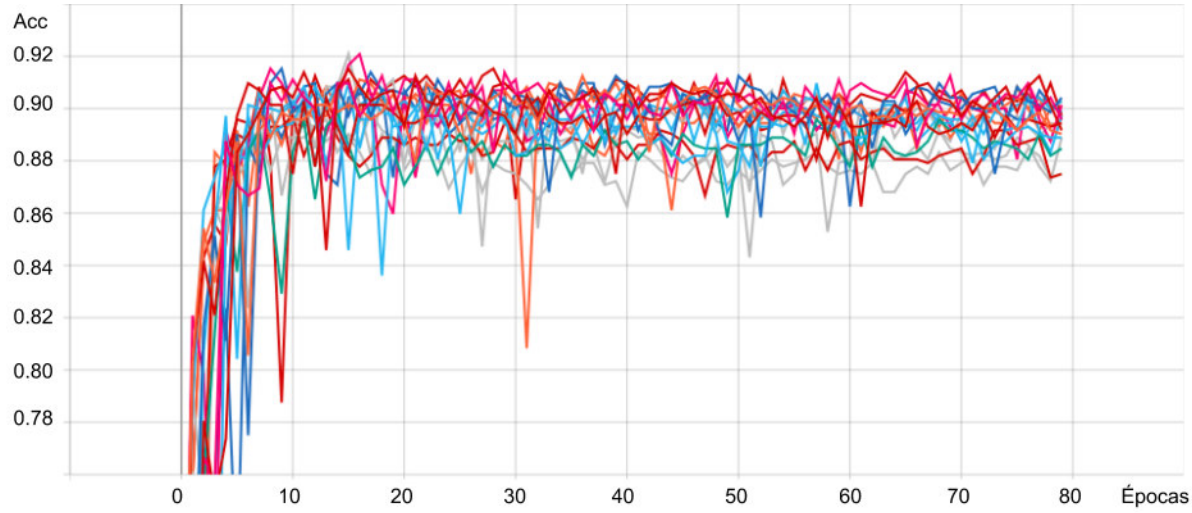
Fonte: O Autor.

Figura 4.2 – Custo das redes no conjunto de Treinamento.



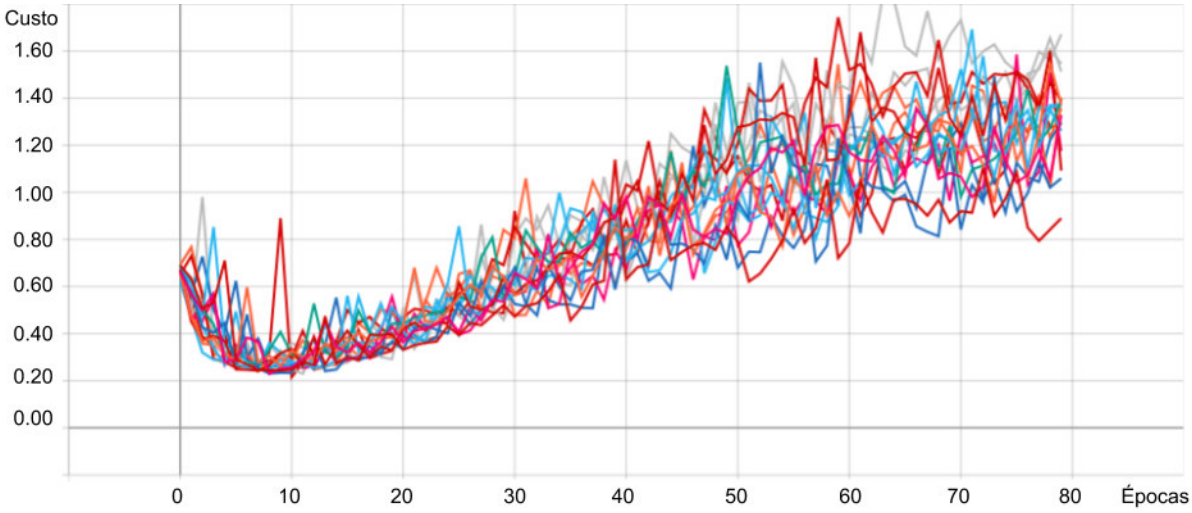
Fonte: O Autor.

Figura 4.3 – Acurácia das redes no conjunto de Validação.



Fonte: O Autor.

Figura 4.4 – Custo das redes no conjunto de Validação.



Fonte: O Autor.

É possível notar que as redes conseguiram aprender bem os dados de treinamento, todas estabilizando-se com acurácia de 100%, para os dados de validação houve uma variação na acurácia das redes embora seja possível notar a tendencia global das mesmas em convergir para 90%. No entanto para os resultados de custo no conjunto de validação é possível notar um comportamento no aprendizado, inicialmente há uma queda, e depois os valores voltam a subir, demonstrando que as redes estão super-ajustando ao conjunto de treinamento, é possível notar também que a acurácia estabiliza, demonstrando que a rede não está mais aprendendo.

Observando os valores, buscou-se as três melhores redes que tivessem a melhor relação acurácia-custo, deseja-se que as redes estejam no vale do gráfico de custo da validação e, ao mesmo tempo, antes da estabilização dos valores de acurácia. A busca foi feita manualmente observando os valores de acurácia por época de treinamento a fim de escolher as três melhores redes. A Tabela 4.2 apresenta as três redes que tiveram o melhor desempenho com suas respectivas arquiteturas descritas na Tabela 3.6 e suas épocas.

Tabela 4.2 – Arquitetura e desempenho das três melhores redes.

Rede	***Embedding***	**Dimensão**	**LSTM**	**Dense**	**Época**	**Acurácia**	**Custo**
1	128	64	3	1	11	90,14%	**0,2156**
2	128	128	1	1	9	90,97%	0,2299
3	128	128	3	1	16	**92,08%**	0,2785

Obtida as três redes, o próximo passo é verificar o seu desempenho no conjunto de dados de Teste para comparação com os resultados da Tabela 4.1. A Tabela 4.3 apresenta o desempenho das três redes nos dados de Teste.

Tabela 4.3 – Resultados utilizando a Abordagem *B*.

Algoritmo	**Acurácia**	**Precisão**	**Recall**	**Especificidade**
Rede 1	**92,36%**	**0,9446**	0,9	**0,9472**
Rede 2	90,28%	0,8940	**0,9139**	0,8917
Rede 3	91,80%	0,9337	0,9	0,9361

É possível observar que a rede 1 se sobressaiu nos resultados atingindo a acurácia de 92,36% enquanto que as redes 1 e 2 se atingiram 90,28% e 91,80% respectivamente, e assim, obtiveram valores um pouco mais altos que o obtido pelo SVM na Tabela 4.1. Foi possível observar uma leve melhoria dos algoritmos de DL em comparação com o SVM, demonstrando que seus nós conseguiram generalizar bem obtendo os padrões das *Fake News*.

O desempenho das redes já era esperado ser maior que o dos algoritmos tradicionais pois a Abordagem *B* considera a ordem das palavras através das camadas LSTM, um fator bem relevante quando se trata de mineração de texto, onde várias frases podem ser formadas utilizando as mesmas palavras. No entanto, o desempenho do algoritmo SVM foi apenas um pouco abaixo, demonstrando que o método TF-IDF também consegue obter informações relevantes em textos e o algoritmo consegue generalizar bem dados com alta dimensionalidade.

Outro fator importante é o desempenho das três redes e do SVM na precisão, *recall* e especificidade. Considerando que é importante ser mais preciso sobre o que é *Fake News* e não marcar incorretamente uma notícia verdadeira como falsa, é necessário observar

o valor da precisão. Quanto menor a precisão, significa que o classificador marcou mais notícias reais como *Fake News*.

Levando isso em consideração, a rede neural 1 teve um bom desempenho, pois em 94,46% das vezes que marcou algo como *Fake News*, ela acertou, enquanto que o SVM teve alto valor de *recall* demonstrando que acertou bem os exemplos da classe *Fake News* porém o baixo valor de especificidade evidencia que muitas notícias reais foram marcadas como falsas, o que não é desejável.

5 Conclusão

Preocupado com a gravidade que o tema *Fake News* apresenta à nossa sociedade, este trabalho propôs a detecção de FN, em meio a notícias verdadeiras, utilizando algoritmos de inteligência artificial. Foram estes os algoritmos: Árvore de Decisão, *Naive Bayes*, SVM e Redes Neurais Profundas, divididos em duas abordagens.

A abordagem *A* utiliza a criação de uma *bag of words* com, com todas as palavras que ocorreram no conjunto de treinamento de frequência mínima três e utiliza a técnica TF-IDF para e avaliar a importância de uma palavra contida no texto. Já a abordagem *B*, que também utiliza a *bag of words* com frequência mínima três palavras, emprega uma técnica diferente, a *Word Embedding*, que organiza cada palavra da *bag of words* por índices em um vetor.

O treinamento foi realizado dividindo as abordagens. A abordagem *A* para os algoritmos clássicos de aprendizagem de máquina, Árvore de Decisão, *Naive Bayes* e SVM, enquanto que a Abordagem *B* utilizou as redes neurais profundas. O conjunto total possuía 7200 dados, destes, 5760 foram utilizados para treinamento, 720 para validação e 720 para teste. Vale ressaltar que, ainda para a abordagem *B*, 36 arquiteturas de redes foram treinadas com o conjunto de treino e testadas com o conjunto de validação. Todas foram treinadas por 80 gerações, pois a partir daí não obtiveram melhoras no desempenho.

Os resultados da abordagem *A* mostram que o SVM se sobressaiu entre os demais obtendo acurácia de 90,13% enquanto Árvore de Decisão e *Naive Bayes* obtiveram acurácia abaixo dos 80%, com 77,77% e 71,25% respectivamente, além disso obtiveram um *recall* bem distante do SVM, constituindo valores abaixo de 0,7000. Esses último resultado, em especial, mostra que os algoritmos Árvore de Decisão, *Naive Bayes* apresentaram dificuldade de predizer quais dados realmente eram *Fake News* (verdadeiro-positivos).

Com a abordagem *B* foi possível encontrar uma rede que obteve desempenho de 92,36% de acurácia e 94,46% de precisão, apresentando uma melhoria em comparação com os outros resultados. Essa melhoria é mais evidente quando se compara os valores de precisão e especificidade, dos quais as redes demonstraram não predizer erroneamente notícias reais como falsas, fato que é mais desejável para o problema de detecção de *Fake News*

Este trabalho fica como uma contribuição aos resultados obtidos do trabalho de Monteiro et al. (2018) com novos resultados devido ao uso de DL com *Word2Vec* e LSTM.

Para trabalhos futuros, fica a proposta de utilizar técnicas mais sofisticadas de treinamento de redes neurais profundas e de escolha de arquitetura que tenha encaixe

melhor para o domínio que está sendo tratado.

Referências

ABRAHAM, A. Artificial neural networks. *handbook of measuring system design*, Wiley Online Library, 2005. Citado 3 vezes nas páginas 22, 23 e 24.

ALBERTO, A. *Ética e códigos da comunicação social*. [S.l.]: Editora SULINA, 1970. v. 10. Citado na página 10.

APRÁ, A. *ELevantamento feito com dados da USP embasa lista dos 10 maiores sites de "falsas notícias"no Brasil. 2017*. 2017. Disponível em: <https://www.issoenoticia.com.br/artigo/projeto-da-usp-lista-10-maiores-sites-de-falsas-noticias-no-brasil>. Acesso em: 29 de agosto de 2018. Citado na página 12.

AURÉLIO, B. de H. F. Novo dicionário da língua portuguesa. In: *Novo dicionário da língua portuguesa*. [S.l.]: Nova Fronteira, 1986. Citado na página 10.

BARION, E. C. N.; LAGO, D. Mineração de textos. *Revista de Ciências Exatas e Tecnologia*, v. 3, n. 3, p. 123–140, 2015. Citado na página 13.

BECKER, K.; TUMITAN, D. Introdução à mineração de opiniões: Conceitos, aplicações e desafios. *Simpósio Brasileiro de Banco de Dados*, 2013. Citado na página 13.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural language processing with Python*. [S.l.]: "O'Reilly Media, Inc.", 2009. Citado na página 28.

BRAGA, A. d. P. *Redes neurais artificiais: teoria e aplicações*. [S.l.]: Livros Técnicos e Científicos, 2000. Citado na página 24.

CHERKASSKY, V.; MULIER, F. Vapnik-chervonenkis (vc) learning theory and its applications. *IEEE Transactions on Neural Networks*, IEEE Computational Intelligence Society, v. 10, n. 5, p. 985–987, 1999. Citado 2 vezes nas páginas 18 e 19.

CONTRIBUTORS, W. *Spamming - Wikipedia, The Free Encyclopedia*. 2019. Disponível em: <https://en.wikipedia.org/w/index.php?title=Spamming&oldid=905493027>. Acesso em: 10 de Julho de 2019. Citado na página 12.

Data Science Academy. *Deep Learning Book*. 2019. Disponível em: <<http://www.deeplearningbook.com.br/>>. Acesso em: Acesso em: 10 de julho de 2019. Citado na página 25.

DENG, L.; YU, D. *Deep Learning: Methods and Applications*. [S.l.], 2014. Disponível em: <https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/>. Citado na página 25.

FACELI, K. et al. Inteligência artificial: Uma abordagem de aprendizado de máquina. *Rio de Janeiro: LTC*, 2011. Citado 7 vezes nas páginas 14, 15, 16, 17, 20, 21 e 22.

FACELI, K. et al. Inteligência artificial: Uma abordagem de aprendizado de máquina. 2011. Citado na página 19.

GOLDBERG, Y.; LEVY, O. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014. Citado na página 14.

GRANIK, M.; MESYURA, V. Fake news detection using naive bayes classifier. In: IEEE. *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*. [S.l.], 2017. p. 900–903. Citado na página 12.

GURNEY, K. An introduction to neural networks. 1997. Citado na página 22.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2007. Citado na página 22.

JR, E. C. T.; LIM, Z. W.; LING, R. Defining "fake news" a typology of scholarly definitions. *Digital journalism*, Taylor & Francis, v. 6, n. 2, p. 137–153, 2018. Citado 2 vezes nas páginas 10 e 12.

KOVALESKI, P. de A. *Implementação de Redes Neurais Profundas para Reconhecimento de Ações em Vídeos*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2018. Citado na página 23.

LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007. Citado na página 17.

MARUMO, F. S. Deep learning para classificação de fake news por sumarização de texto. 2018. Citado 6 vezes nas páginas 12, 13, 14, 24, 26 e 27.

MENDES, M. d. S. Aprendizado em profundidade na descrição semântica de imagens. 2018. Citado na página 26.

MITCHELL, T. M. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, v. 45, p. 37, 1997. Citado 2 vezes nas páginas 17 e 20.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas Inteligentes-Fundamentos e Aplicações*, v. 1, n. 1, 2003. Citado na página 17.

MONTEIRO, R. A. et al. Contributions to the study of fake news in portuguese: New corpus and automatic detection results. In: *Computational Processing of the Portuguese Language*. [S.l.]: Springer International Publishing, 2018. p. 324–334. ISBN 978-3-319-99722-3. Citado 3 vezes nas páginas 28, 34 e 39.

MORAIS, E. A. M.; AMBRÓSIO, A. P. L. Mineração de textos. *Relatório Técnico–Instituto de Informática (UFG)*, 2007. Citado na página 13.

OLAH, C. *Understanding LSTM Networks – colah's blog*. 2015. Disponível em: <<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>>. Acesso em: Acesso em: 10 de julho de 2019. Citado 2 vezes nas páginas 26 e 27.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 31.

PIMENTA, A. et al. Weka-g: mineração de dados paralela em grades computacionais. *Revista de Sistemas de Informação da FSMA*, v. 4, p. 2, 2009. Citado na página 16.

RUEDIGER, M. A. et al. Robôs, redes sociais e política no brasil: estudo sobre interferências ilegítimas no debate público na web, riscos à democracia e processo eleitoral de 2018. FGV DAPP, 2017. Citado na página 12.

Samarth Agrawal, Towards Data Science. *Sentiment Analysis using LSTM (Step-by-Step Tutorial) - Using PyTorch framework for Deep Learning*. 2019. Disponível em: <https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948>. Acesso em: 07 de julho de 2019. Citado na página 33.

SANTOS, F. L. d. Mineração de opinião em textos opinativos utilizando algoritmos de classificação. 2014. Citado na página 15.

SANTOS, F. L. d. Mineração de opinião em textos opinativos utilizando algoritmos de classificação. 2014. Citado na página 18.

SOUSA, G. G. B. *Deep Learning para a Detecção e Classificação de Pneumonia por Radiografias do Tórax*. 44 p. Monografia (Graduação) — Ciências da Computação, Universidade Federal do Maranhão, São Luís, 2018. Citado na página 23.

TensorFlow: library. *TensorFlow*. 2019. Disponível em: <https://www.tensorflow.org/install>. Acesso em: 07 de julho de 2019. Citado na página 31.

TSYTSARAU, M.; PALPANAS, T. Survey on mining subjective data on the web. *Data Mining and Knowledge Discovery*, Kluwer Academic Publishers, v. 24, n. 3, p. 478–514, 2012. Citado na página 14.

VAPNIK, V. *The nature of statistical learning theory*. [S.l.]: Springer science & business media, 2013. Citado na página 17.

VAPNIK, V. N.; CHERVONENKIS, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. In: *Measures of complexity*. [S.l.]: Springer, 2015. p. 11–30. Citado na página 17.

WITTEN, I. H.; FRANK, E. *Data Mining: Practical machine learning tools and techniques*. 2. ed. San Francisco: Morgan Kaufmann, 2005. Citado na página 21.

WITTEN, I. H. et al. *Data Mining: Practical machine learning tools and techniques*. [S.l.]: Morgan Kaufmann, 2016. Citado na página 16.