

**UNIVERSIDADE FEDERAL FLUMINENSE**  
**INSTITUTO DE COMPUTAÇÃO**  
**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**PHELIPE GONÇALVES MARTINS**

**AVALIAÇÃO DE ABORDAGENS BASEADAS EM DEEP LEARNING PARA A  
IDENTIFICAÇÃO DE FAKE NEWS**

**NITERÓI – RJ**

**2018**

Ficha catalográfica automática - SDC/BEE  
Gerada com informações fornecidas pelo autor

M379a    Martins, Phelipe Gonçalves  
          Avaliação de Abordagens Baseadas em Deep Learning para a  
          Identificação de Fake News : / Phelipe Gonçalves Martins ;  
          José Viterbo Filho, orientador ; Eduardo de Oliveira Andrade,  
          coorientador. Niterói, 2018.  
          75 f.

          Trabalho de Conclusão de Curso (Graduação em Ciência da  
          Computação)-Universidade Federal Fluminense, Escola de  
          Engenharia, Niterói, 2018.

          1. Aprendizado de máquina. 2. Inteligência artificial. 3.  
          Produção intelectual. I. Viterbo Filho, José, orientador.  
          II. Andrade, Eduardo de Oliveira, coorientador. III.  
          Universidade Federal Fluminense. Escola de Engenharia. IV.  
          Título.

CDD -

**PHELIPE GONÇALVES MARTINS**

**AVALIAÇÃO DE ABORDAGENS BASEADAS EM DEEP LEARNING PARA A  
IDENTIFICAÇÃO DE FAKE NEWS**

**Monografia apresentada ao  
Departamento de Ciência da  
Computação da Universidade Federal  
Fluminense, como requisito para  
obtenção do Grau de Bacharel em  
Ciência da Computação.**

**Orientador: Prof. José Viterbo Filho**

**Coorientador: Eduardo de Oliveira Andrade**

**NITERÓI – RJ**

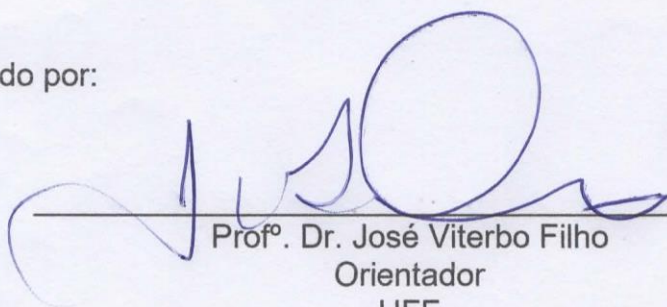
**2018**

**PHELIPE GONÇALVES MARTINS**

**AVALIAÇÃO DE ABORDAGENS BASEADAS EM DEEP LEARNING PARA  
A IDENTIFICAÇÃO DE FAKE NEWS**

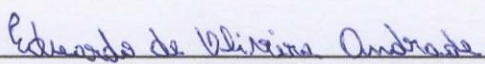
**Monografia apresentada ao  
Departamento de Ciência da  
Computação da Universidade Federal  
Fluminense, como requisito para  
obtenção do Grau de Bacharel em  
Ciência da Computação.**

Aprovado por:



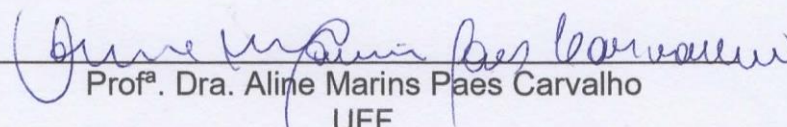
---

Prof. Dr. José Viterbo Filho  
Orientador  
UFF



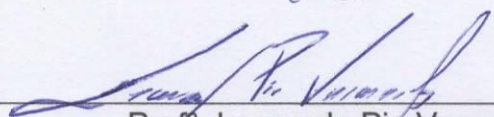
---

Eduardo de Oliveira Andrade  
Coorientador  
UFF



---

Prof. Dra. Aline Marins Paes Carvalho  
UFF



---

Prof. Leonardo Pio Vasconcelos  
UFF

**NITERÓI - RJ**

**2018**

Dedico este trabalho a todos aqueles que buscam conhecimento, pois isto é a base da nossa sociedade.

## **AGRADECIMENTOS**

Agradeço, primeiramente, a minha namorada Stéphanie R. Alves, que me acompanha há oito anos e que me deu força e apoio para sempre seguir em frente. A minha mãe Janete G. Martins, meu pai Fernando de O. Martins e meu irmão Rodrigo G. Martins, que são a minha base e que me proporcionaram condições para que eu ingressasse no ensino superior. Aos amigos Vinicius B. Nogueira, Ramon A. Estevez, Guilherme N. Farias e Pedro G. S. Miranda com quem sempre puder contar. Ao meu orientador José V. Filho e ao meu coorientador Eduardo de O. Andrade que de bom grado, aceitaram o desafio de me guiar na realização deste trabalho. Por último, mas não menos importante, agradeço a todos aqueles que sempre torceram por mim.

## RESUMO

Nos últimos anos, um fenômeno social tem ganhado destaque no mundo todo, por seu alcance e rápida divulgação entre as pessoas: são as chamadas *fake news*. Este tipo de notícia já demonstrou em diversas situações a capacidade que possui de causar impactos negativos a sociedade. Dessa forma, diversos estudos têm sido realizados para que se ache alguma solução viável para combater diretamente tal fenômeno. Recentemente, o uso de técnicas de Aprendizado de Máquina tem se mostrado uma alternativa interessante para apontar se uma dada notícia é falsa ou não. Contudo, devido a uma grande quantidade de algoritmos existentes nesta subárea de inteligência artificial, é importante que testes sejam realizados para que se verifique quais destes são mais apropriados para esta tarefa. Este trabalho segue uma abordagem experimental e exploratória, no intuito de avaliar qual método de abordagem tradicional (*Multinomial Naive Bayes* e *Support Vector Machines*) e de Deep Learning (*Convolutional Neural Network* e *Long Short-Term Memory*) possuem a melhor performance. Além disso, verificar qual método dentre esses dois produz melhores resultados na identificação de *fake news*, utilizando os parâmetros ideais e três bases de dados. As mesmas contendo mil, dez mil e cem mil notícias classificadas como 'FAKE' e 'REAL'. Após diversos testes, verificou-se que *CNN* possui a melhor performance em todas as bases de dados, destacando-se a acurácia de 97,09% e 98,39% para as de dez mil e cem mil notícias, respectivamente. Ainda, *SVM* demonstrou níveis de precisão abaixo das *CNN*, porém muito interessantes, com valores maiores ou iguais a 90% desde que se use *Term frequency-inverse document frequency*. *LSTM* demonstrou que vale a pena ser utilizada somente com a base de dados contendo cem mil notícias com acurácia de 96,59%. Já para *MNB*, constatou-se que não vale a pena ser utilizado para a resolução do problema proposto.

Palavras-chave: Aprendizado de Máquina, *Multinomial Naive Bayes*, *Support Vector Machine*, *Convolutional Neural Network*, *Long Short-Term Memory*.

## ABSTRACT

In recent years, a social phenomenon has been prominent worldwide, because of its reach and fast dissemination among people: they are called fake news. This type of news has already demonstrated in several situations the capacity it has of causing negative impacts to society. In this way, several studies have been carried out to find some viable solution to directly combat this phenomenon. Recently, the use of Aprendizado de Máquina techniques has been an interesting alternative to point out whether a given news is fake or not. However, due to a large number of algorithms in this artificial intelligence subarea, it is important that tests are performed to determine which of these are most appropriate for this task. This work follows an experimental and exploratory approach, in order to evaluate which traditional approach method (Multinomial Naive Bayes and Support Vector Machines) has better performance, which method of Deep Learning approach (Convolutional Neural Network and Long Short-Term Memory), which is a Aprendizado de Máquina subarea that is on the rise, also has better performance and which method among these two produces better results in the identification of fake news, using the ideal parameters and three databases containing one thousand, ten thousand and one hundred thousand news classified as 'FAKE' and 'REAL'. After several tests, it was verified that CNN has the best performance in all the databases, highlighting the accuracy of 97.09% and 98.39% for the ten thousand and one hundred thousand news, respectively. Still, SVM demonstrated accuracy levels below CNN, but very interesting, with values greater than or equal to 90% since the use of the term frequency-inverse document frequency. LSTM has shown that it is worth using only with the database containing one hundred thousand news with accuracy of 96.59%. As for MNB, it was found that it is not worth being used to solve the problem proposed.

Keywords: Aprendizado de Máquina, Multinomial Naive Bayes, Support Vector Machine, Convolutional Neural Network, Long Short-Term Memory.



## LISTA DE FIGURAS

Figura 1 - Exemplo de classificação binária .....	8
Figura 2 - Exemplo de palavras de parada .....	11
Figura 3 - Hiperplanos que dividem com sucesso o classificador SVM .....	16
Figura 4 - Hiperplano ótimo - SVM.....	17
Figura 5 – Contexto no qual Deep Learning está inserido .....	18
Figura 6 - Convolução 1D .....	20
Figura 7 - Rede Neural de Recorrência.....	22
Figura 8 - Estrutura de camadas para CNN com dataset pequeno.....	29
Figura 9 - Estrutura de camadas para LSTM com dataset pequeno .....	29
Figura 10 - MNB com BoW e 1.000 dados: matriz de confusão.....	32
Figura 11 - MNB com BoW e 10.000 dados: matriz de confusão.....	33
Figura 12 - MNB com BoW e 100.000 dados: matriz de confusão.....	35
Figura 13 - MNB com Tf-idf e 1.000 dados: matriz de confusão .....	36
Figura 14 - MNB com Tf-idf e 10.000 dados: matriz de confusão .....	37
Figura 15 - MNB com Tf-idf e 100.000 dados: matriz de confusão .....	39
Figura 16 - SVM com BoW e 1.000 dados: matriz de confusão.....	40
Figura 17 - SVM com Bow e 10.000 dados: matriz de confusão.....	41
Figura 18 - SVM com BOW e 100.000 dados: Matriz de confusão .....	43
Figura 19 - SVM com Tf-idf e 1.000 dados: matriz de confusão.....	44
Figura 20 - SVM com Tf-idf e 10.000 dados: matriz de confusão.....	45
Figura 21 - SVM com Tf-idf e 100.000 dados: matriz de confusão.....	47
Figura 22 - CNN com 1.000 dados: matriz de confusão.....	49

Figura 23 - CNN com 10.000 dados: matriz de confusão.....	50
Figura 24 - CNN com 100.000 dados: matriz de confusão.....	51
Figura 25 - LSTM com 1.000 dados: matriz de confusão .....	53
Figura 26 - LSTM com 10.000 dados: matriz de confusão .....	54
Figura 27 - LSTM com 100.000 dados: matriz de confusão .....	55

## LISTA DE TABELAS

Tabela 1 - Exemplo de tokenização .....	10
Tabela 2 - Matriz Documento-Termo ( <i>Bag of Words</i> ).....	13
Tabela 3 - Matriz Documento-Termo ( <i>Tf-idf</i> ) .....	14
Tabela 4 - Dados dos datasets.....	26
Tabela 5 – Estrutura da base de dados utilizada .....	26
Tabela 6 - Relação de acurácias de acordo com o <i>dataset</i> .....	56

## LISTA DE GRÁFICOS

Gráfico 1 - <i>MNB</i> com <i>BoW</i> e 1.000 dados: <i>alphas x scores</i> .....	31
Gráfico 2 - <i>MNB</i> com <i>BoW</i> e 10.000 dados: <i>alphas x scores</i> .....	33
Gráfico 3 - <i>MNB</i> com <i>BoW</i> e 100.000 dados: <i>alphas x scores</i> .....	34
Gráfico 4 - <i>MNB</i> com <i>Tf-idf</i> e 1.000 dados: <i>alphas x scores</i> .....	35
Gráfico 5 - <i>MNB</i> com <i>Tf-idf</i> e 10.000 dados: <i>alphas x scores</i> .....	37
Gráfico 6 - <i>MNB</i> com <i>Tf-idf</i> e 100.000 dados: <i>alphas x scores</i> .....	38
Gráfico 7 - <i>SVM</i> com <i>BoW</i> e 1.000 dados: <i>Cs X scores</i> .....	39
Gráfico 8 - <i>SVM</i> com <i>BOW</i> e 10.000 dados: <i>Cs x scores</i> .....	41
Gráfico 9 - <i>SVM</i> com <i>BoW</i> e 100.000 dados: <i>Cs X scores</i> .....	42
Gráfico 10 - <i>SVM</i> com <i>Tf-idf</i> e 1.000 dados: <i>Cs x scores</i> .....	43
Gráfico 11 - <i>SVM</i> com <i>Tf-idf</i> e 10.000 dados: <i>Cs x scores</i> .....	45
Gráfico 12 - <i>SVM</i> com <i>Tf-idf</i> e 100.000 dados: <i>Cs x scores</i> .....	46
Gráfico 13 - <i>CNN</i> com 1.000 dados: gráficos de acurácia e perda .....	48
Gráfico 14 - <i>CNN</i> com 10.000 dados: gráficos de acurácia e perda .....	49
Gráfico 15 - <i>CNN</i> com 100.000 dados: gráficos de acurácia e perda .....	51
Gráfico 16 - <i>LSTM</i> com 1.000 dados: gráficos de acurácia e perda.....	52
Gráfico 17 - <i>LSTM</i> com 10.000 dados: gráficos de acurácia e perda.....	53
Gráfico 18 - <i>LSTM</i> com 100.000 dados: gráficos de acurácia e perda.....	55

## SUMÁRIO

<b>AGRADECIMENTOS .....</b>	<b>V</b>
<b>RESUMO.....</b>	<b>VI</b>
<b>ABSTRACT.....</b>	<b>VII</b>
<b>1. INTRODUÇÃO .....</b>	<b>1</b>
1.1. DEFINIÇÃO DO PROBLEMA.....	3
1.2. OBJETIVOS .....	4
<b>1.2.1. OBJETIVOS ESPECÍFICOS.....</b>	<b>5</b>
<b>2. CONCEITOS BÁSICOS .....</b>	<b>6</b>
2.1. APRENDIZADO DE MÁQUINA .....	6
2.2. APRENDIZADO SUPERVISIONADO.....	6
2.3. CLASSIFICAÇÃO.....	7
2.4. PROCESSAMENTO DE LINGUAGEM NATURAL.....	8
<b>2.4.1. PRÉ-PROCESSAMENTO DOS DADOS .....</b>	<b>9</b>
<b>2.4.2. EXTRAÇÃO DE CARACTERÍSTICAS DE TEXTOS .....</b>	<b>12</b>
2.5. CLASSIFICAÇÃO <i>NAIVE BAYES</i> .....	15
2.6. CLASSIFICAÇÃO <i>SUPPORT VECTOR MACHINES</i> KERNEL LINEAR.....	16
2.7. DEEP LEARNING .....	17
2.8. REDE NEURAL DE CONVOLUÇÃO.....	19
2.9. REDE NEURAL DE RECORRÊNCIA.....	22
2.10. LONG SHORT-TERM MEMORY.....	23
2.11. TRABALHOS RELACIONADOS.....	23
<b>3. METODOLOGIA .....</b>	<b>25</b>
3.1. CONFIGURAÇÃO DAS PLATAFORMAS DE EXECUÇÃO.....	25
3.2. BASES DE DADOS.....	26

3.3.	EXECUÇÃO DA METODOLOGIA.....	27
<b>4.</b>	<b>ANÁLISE EXPERIMENTAL .....</b>	<b>31</b>
4.1.	ABORDAGEM TRADICIONAL .....	31
4.1.1.	MULTINOMIAL NAIVE BAYES COM BAG-OF-WORDS .....	31
4.1.2.	MULTINOMIAL NAIVE BAYES COM TF-IDF .....	35
4.1.3.	SUPPORT VECTOR MACHINES COM BOW .....	39
4.1.4.	SUPPORT MACHINE VECTORS COM TF-IDF.....	43
4.2.	ABORDAGEM DEEP LEARNING .....	47
4.2.1.	CONVOLUTIONAL NEURAL NETWORK.....	48
4.2.2.	LONG SHORT-TERM MEMORY .....	52
4.3.	COMPARAÇÃO DOS RESULTADOS .....	56
<b>5.</b>	<b>CONCLUSÃO .....</b>	<b>58</b>
5.1.	LIMITAÇÕES.....	58
5.2.	TRABALHOS FUTUROS .....	59
	<b>REFERÊNCIAS BIBLIOGRÁFICAS.....</b>	<b>60</b>

## 1. INTRODUÇÃO

Ao longo do século XXI, o número de usuários que utilizam e possuem internet em suas casas tem crescido bastante, ano após ano. “Estima-se que, em 2015, 43,4% de indivíduos em todo o mundo estão online [...]. A UIT prevê que 53 por cento de indivíduos em todo o mundo estarão usando a Internet até 2020” (ITU, 2015, tradução nossa). Não é incomum o fato de que a porcentagem de usuários cresceu de maneira tão rápida, visto que a internet é um grande facilitador no nosso dia-a-dia. Fica evidente, diante desse quadro que esse número irá aumentar ainda mais.

Um de seus principais usos é a busca por informação, seja através de redes sociais, sites de buscas, blogs, *podcasts* ou qualquer outro meio no qual se consiga conhecimento. Dessa maneira, é importante ressaltar que:

A PBM do ano de 2016 verifica que a rede mundial de computadores se cristaliza como segunda opção dos brasileiros na busca de informação, atrás somente da televisão. Quase a metade dos brasileiros (49%) declarou usar a web para obter notícias (primeira e segunda menções), percentual abaixo da TV (89%), mas bem acima do rádio (30%), dos jornais (12%) e das revistas (1%). (SECOM, 2016).

Conforme citado acima, aqui no Brasil a internet está se solidificando cada vez mais, mostrando que novas tecnologias estão superando meios antigos de informação como o rádio e jornais. Logo, é de se esperar que a internet está sendo massivamente usada como fonte de informação, não só pela facilidade, mas também pela sua utilidade.

Ultimamente, o fenômeno social conhecido como *Fake News* tem ganhado bastante destaque. De acordo com Porcello e Brites (2018) em relação às *Fake News*:

[...] as fakenews não são notícias distorcidas, erradas ou mal apuradas. Elas são notícias falsas criadas propositalmente para enganar visando alguma vantagem sobre isso. Os boatos sempre existiram, o que muda é o contexto em que estamos inseridos, a velocidade e a profissionalização com que as fakenews tem se multiplicado para atingir um numero cada vez maior de pessoas.

De acordo essa explicação, um dos motivos para as *Fake News* se multiplicarem são a velocidade e o contexto que estamos inseridos. A internet é um grande facilitador, e dessa forma ficou mais fácil gerar notícias falsas e espalhá-las. Logo, o uso indevido de tecnologias pode prejudicar nossas vidas e afetar fatos históricos.

A disseminação de notícias falsas não é uma novidade, porém com a popularização das redes sociais, o número de divulgação desse tipo de notícia, que é potencializado através do compartilhamento da mesma pelos usuários, aumentou drasticamente. Isto acontece devido ao teor das notícias, que geralmente possuem temas que são relevantes para a população, influenciando o seu compartilhamento entre conhecidos (DFNDR LAB, 2018). A divulgação de notícias falsas na grande maioria das vezes é relacionada a temas que afetam diretamente a população, fazendo com que constantemente pessoas divulguem tais notícias apenas lendo o título das mesmas e não o seu conteúdo e fonte. Por todos esses aspectos, não é de se espantar que as *fake news* são divulgadas tão facilmente, por ter todo esse lado influenciador.

Em épocas de eleições, pode-se notar um aumento ainda maior desse fenômeno, que costuma ser utilizado para difamar determinados candidatos, o que aconteceu com bastante frequência no Brasil durante o ano de 2018. “As eleições brasileiras, por sua vez, foram um dos temas que mais contribuíram para o aumento das divulgações de Notícias falsas no terceiro trimestre do ano.” (DFNDR LAB, 2018). Nessa época, tem se tornado comum o fato de que as pessoas tentem impor seus pontos de vista sobre determinados candidatos, e uma das formas de se fazer isso é através de compartilhamento de notícias que influenciem na tomada de decisão.

Tal fenômeno social pode ocorrer de diversas formas, podendo afetar de maneira negativa desde pessoas normais até celebridades e políticos. Em relação a isso:

O mundo virtual acaba por se tornar um lugar onde os ânimos, instigados por preferências partidárias ou inclinações ideológicas, proporcionam um mercado de exercício de controle ou de influência de pessoas, por meio de mentiras e boatos. (CAJÚ, 2017).



Conforme citado acima, pessoas são prejudicadas com a disseminação de *fake news*, fazendo com que mesmo após a divulgação da notícia verdadeira, não diminua o nível de ódio anteriormente disseminado ou a imposição do pensamento, fazendo com que as notícias falsas alcancem os seus objetivos.

Segundo Gomes (2017), as empresas Google e Facebook estão combatendo tal fenômeno, no intuito de frear a dissipação de informações falsas, que prejudicam a população, através da penalidade financeira para aqueles que criam tais conteúdos. Como esse fenômeno tem se mostrado extremamente prejudicial e interessante ao mesmo tempo, e com o avanço da computação de modo geral, diversas empresas ao redor do mundo tem investido em tecnologia no combate ao problema das *Fake News*, geralmente utilizando técnicas de *Aprendizado de Máquina* (ML) e Processamento de Linguagem Natural (PLN), propiciando que uma ferramenta computacional determine se uma determinada notícia é falsa ou verdadeira, com um nível de precisão satisfatório. “Tecnologias como Inteligência Artificial (IA) e Processamento de Linguagem Natural (PLN) oferecem grandes promessas para os pesquisadores construírem sistemas que detecta automaticamente notícias falsas” (THOTA *et al.*, 2018, tradução nossa).

Conforme explicado na citação acima, pode-se concluir que o investimento em determinadas tecnologias facilita no processo de resolução desse fenômeno. Isto se dá através de subáreas da inteligência artificial, que estão em alta expansão no momento, chamando a atenção de mais estudiosos.

### 1.1. DEFINIÇÃO DO PROBLEMA

Diversas técnicas e algoritmos podem ser utilizados para a resolução do problema das *Fake News*. É de extrema importância que se analise quais destas retornam resultados desejáveis, ou seja, quais possuem o melhor nível de precisão, dadas determinadas situações, pois uma notícia classificada de maneira errada pode gerar danos à sociedade em geral. Além disso, é importante verificar a quantidade de tempo gasta em cada caso e qual possui melhor desempenho com volume de dados grande, médio e pequeno.

Dentre as variadas abordagens em *Aprendizado de Máquina*, para o problema de determinar uma notícia como sendo verdadeira ou falsa, que é um problema de classificação, ou seja, dada uma entrada é atribuído um rótulo a ela, de caráter distinto como “verdadeiro” ou “falso”, são utilizadas técnicas de aprendizado supervisionado, onde o programa é treinado sobre um conjunto de dados já classificados e a partir daí é possível prever a classificação de dados não classificados, com certo nível de exatidão. Alguns dos modelos comumente utilizados são *Support Vector Machine* (SVM) e *Naive Bayes* (NB). Para esses modelos, assim como para todos os outros existentes em aprendizado supervisionado, é ideal realizar um pré-processamento dos dados, utilizando técnicas de PLN.

Outra técnica que está sendo bastante utilizada ultimamente, devido aos seus resultados promissores é o *Deep Learning*, que é mais uma abordagem de ML, porém essa utiliza redes neurais artificiais profundas no processamento de informações e aprendizagem. Seu grande diferencial para as demais abordagens, consiste no fato de que com *Deep Learning* é possível trabalhar com análise de dados brutos, possibilitando um campo de atuação mais amplo. Alguns modelos utilizados são *Convolutional Neural Network* (CNN) e *Recurrent Neural Network* (RNN). Para esse trabalho, as análises serão feitas sobre os modelos citados anteriormente.

## 1.2. OBJETIVOS

O objetivo desse trabalho consiste em realizar uma comparação experimental entre algoritmos de aprendizado de máquina que seguem uma abordagem tradicional com algoritmos baseados em Deep Learning (*deep learning*) a fim de resolver o problema de identificação de *fake news*. Serão treinados quatro modelos usando os algoritmos de aprendizado de máquina, sendo eles: *Support Vector Machine*, *Naive Bayes*, *Convolutional Neural Network* e *Long Short-Term Memory*.

### 1.2.1. OBJETIVOS ESPECÍFICOS

- Selecionar um *dataset* contendo um conjunto de notícias verdadeiras e falsas que possa ser utilizado no problema de classificação de textos.
- Verificar dentre os parâmetros mais utilizados, quais valores geram melhor acurácia para os classificadores *Naive Bayes* e *Support Vector Machine* (SVM) com os modelos *bag-of-words* (BoW) e *term frequency-inverse document frequency* (tf-idf).
- Realizar uma comparação com base em acurácia utilizando os classificadores *Naive Bayes* e *Support Vector Machine* (SVM) com os modelos *bag-of-words* (BoW) e *term frequency-inverse document frequency* (tf-idf).
- Verificar de acordo com os parâmetros mais utilizados, quais valores geram melhor acurácia para a classe de rede neural artificial *Long Short-Term Memory* (LSTM) e *Convolutional Neural Network* (CNN).
- Realizar uma comparação com base em acurácia utilizando as classes de redes neurais artificiais *Long Short-Term Memory* (LSTM) e *Convolutional Neural Network* (CNN).
- Realizar uma comparação com base em acurácia entre os todos os classificadores e classes de redes neurais utilizados.
- Verificar qual classificador ou classe de rede neural possui melhor acurácia com um *dataset* contendo mil, dez mil e cem mil dados.

### 1.3. ORGANIZAÇÃO DO TEXTO

Os capítulos seguintes estão organizados da seguinte maneira: o capítulo 2 introduz os conceitos básicos necessários para a realização do experimento. O mesmo introduz conceitos sobre Aprendizado de Máquina, Aprendizado Supervisionado, Processamento de Linguagem Natural, *Naive Bayes*, *Support Vector Machines*, Deep Learning, Rede Neural de Convolução e Rede Neural de Recorrência. O capítulo 3 aborda sobre os trabalhos relacionados. O capítulo 4 diz respeito à metodologia utilizada para a realização do experimento. Já no capítulo 5 são apresentados os resultados e a análise em cima dos mesmos. Para finalizar, no capítulo 6 é feita a conclusão e trabalhos futuros.

## 2. CONCEITOS BÁSICOS

### 2.1. APRENDIZADO DE MÁQUINA

Podia parecer algo impossível de se imaginar a poucas décadas atrás a idéia de que um computador poderia “aprender” e através desse aprendizado, executar ações a fim de que a melhor decisão fosse tomada, caracterizando-se assim o termo Aprendizado de Máquina (*Aprendizado de Máquina*). “Aprendizado de Máquina consiste em desenvolver algoritmos de previsões eficientes e precisas.” (MOHRI *et al.*, 2012, tradução nossa). Esses algoritmos são de extrema importância, pois através deles é possível facilitar a tomada de decisão, como é feito em bancos e operadoras na detecção de fraudes. Logo, não é difícil de imaginar o porquê cada vez mais empresas estão investindo nessa área, visto que seus algoritmos ajudam bastante não só a protegê-las, mas também a encontrar padrões que facilitem na hora de executar uma ação.

A idéia central por trás desse termo resume-se em um conjunto de algoritmos capazes de analisar dados e extrair informações úteis a partir deles, facilitando a tomada de decisão. Sobretudo, envolve a construção de modelos matemáticos que são os suportes para facilitar o entendimento desses elementos, no qual um programa pode “aprender” através deles, dados alguns parâmetros para os modelos (MURPHY, 2002). Através do processamento de dados, os modelos de Aprendizado de Máquina conseguem enxergar padrões que podem ser difíceis de um ser humano detectar, por se tratarem muitas vezes de dados complexos e também pela grande quantidade deles.

### 2.2. APRENDIZADO SUPERVISIONADO

O aprendizado supervisionado é uma das categorias principais de Aprendizado de Máquina, onde o modelo tenta encontrar padrões que relacionam as entradas dos algoritmos com suas saídas, e em seguida, utilizando novos dados que não estão com suas *saídas*, é possível, utilizando o modelo que encontrou um determinado padrão, ou seja, encontrou uma relação entre a entrada e a saída,

prever quais são suas classificações (*labels*). “O aprendiz recebe um conjunto de exemplos rotulados como dados de treinamento e faz previsões para todos os pontos não vistos.” (MOHRI *et al.*, 2012, tradução nossa). Quanto mais dados tivermos, melhor será o treinamento, pois assim o modelo consegue encontrar um padrão mais preciso. O aprendizado supervisionado pode ser dividido em dois grupos: classificação e regressão. Para este trabalho, parte dele foi feito utilizando aprendizado supervisionado e a sua categoria de classificação.

### 2.3. CLASSIFICAÇÃO

Basicamente, o problema da classificação consiste em, dados um conjunto de atributos (*features*) que estão classificadas, ou seja, estão com suas *labels*, queremos classificar um conjunto de novas entradas. Em relação à classificação, é importante frisar:

[...] o objetivo é aprender o mapeamento das entradas  $\mathbf{x}$  para as saídas  $y$ , onde  $y \in \{1, \dots, C\}$ , com  $C$  sendo o número de classes. Se  $C = 2$ , chamamos de classificação binária (caso em que muitas vezes assumimos  $y \in \{0, 1\}$ ); [...] Assumimos  $y = f(\mathbf{x})$  para alguma função  $f$  desconhecida, e o objetivo do aprendizado é estimar a função  $f$  dado um conjunto de treinamento rotulado, e então fazer previsões usando  $\hat{y} = \hat{f}(\mathbf{x})$ . [...] Nosso objetivo principal é fazer previsões sobre novos insumos, o que significa que são aqueles que não foram vistos anteriormente [...]. (MURPHY, 2012, tradução nossa).

De acordo com a citação acima, em outras palavras, o objetivo da classificação é mapear as entradas em categorias distintas, onde teremos um conjunto de treinamento que vai auxiliar na tarefa de encontrar a função  $f$  e um conjunto de testes que não foi visto ainda pelo modelo, a fim de medir o seu nível de acurácia.

A figura abaixo representa um exemplo de classificação binária, onde temos dois conjuntos. Caso uma nova entrada seja inserida, é necessário encontrar um modelo que verifique em qual conjunto o novo dado deve ser encaixado.

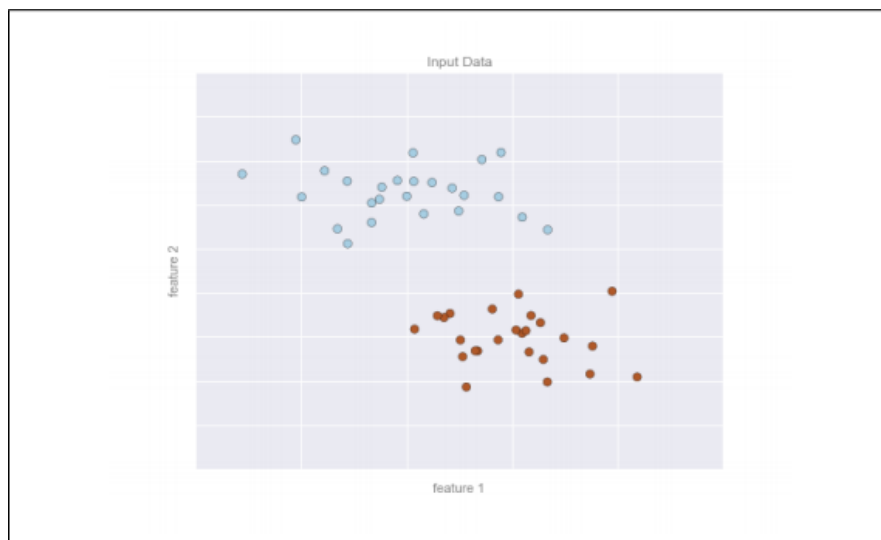


Figura 1 - Exemplo de classificação binária

Fonte: (VANDERPLAS, 2017)

Com isso, evidencia-se que na classificação binária, temos dois conjuntos de dados, e dado uma nova entrada (no caso da figura os círculos amarelos) queremos saber a qual conjunto a mesma pertence. Para a identificação de *Fake News* neste trabalho, a utilização da classificação binária se encaixa perfeitamente, visto que temos algumas bases de dados contendo notícias que estão rotuladas como real ou falsa.

## 2.4. PROCESSAMENTO DE LINGUAGEM NATURAL

A linguagem natural humana é algo fascinante, pois nossa habilidade de desenvolver espontaneamente uma língua bem estruturada, com o intuito de nos comunicarmos dentro de uma comunidade, é algo que somente os seres humanos são capazes de fazer, e diferentemente da linguagem formal, como as linguagens de programação, utilizadas nos computadores, a formalização da linguagem natural não se dá de maneira trivial. “Processamento de linguagem natural também é muito desafiador, pois a linguagem humana é inerentemente ambígua, está sempre mudando e não é bem definida.” (GOLDBERG, 2017, tradução nossa). Dessa forma, desenvolver sistemas computacionais que possam entender nossa língua (falada e escrita) é uma tarefa extremamente complexa. Com esse intuito, surgiu o

Processamento de Linguagem Natural (PLN), que é mais uma das subáreas da Inteligência Artificial que tentam resolver esse problema. Nesse sentido, pode-se constatar:

Processamento de Linguagem Natural (PLN) é o campo de projetar métodos e algoritmos que pegam como entrada ou produz como uma saída não estruturada, dados de linguagem natural [...]. Entender e produzir linguagem usando computadores é portanto altamente desafiador. De fato, o mais conhecido conjunto de métodos para lidar com dados de linguagem é usando aprendizado de máquina supervisionado, que tenta inferir padrões de uso e regularidades a partir de um conjunto de entradas pré-anotadas e pares de saída. (GOLDBERG, 2017, tradução nossa).

Logo, apesar de ser uma tarefa complicada, existem maneiras já estudadas de se trabalhar com PLN. O objetivo é que os computadores consigam interpretar, entender, extrair informações e analisar (de maneira sintática, semântica, léxica e morfológica) textos, através do seu processamento.

#### **2.4.1. PRÉ-PROCESSAMENTO DOS DADOS**

Quando estamos trabalhando com aplicações PLN, existe a necessidade de transformar textos brutos em algo no qual o modelo de *Aprendizado de Máquina* consiga processá-los de maneira melhor, ou seja, é necessário realizar determinados processos sobre dados textuais, a fim de refiná-los, de maneira que eles possam servir da forma mais eficiente possível como dados de entrada, e para tal objetivo é necessário que se faça o pré-processamento dos dados. De maneira precisa, afirma-se:

O pré-processamento de dados é uma das tarefas de mineração de dados que inclui a preparação e a transformação de dados em um formato adequado para o procedimento de mineração. O pré-processamento de dados visa reduzir o tamanho dos dados, localizar as relações entre os dados, normalizar os dados, remover valores discrepantes e extrair recursos para os dados. Inclui várias técnicas como limpeza de dados, integração, transformação e redução. (ALASADI; BHAYA, 2017, tradução nossa).

Logo, quanto mais reduzirmos os ruídos e dados que não são significativos, melhores serão nossas entradas e consequentemente vamos obter melhores resultados. Existem diversas técnicas que podem ser aplicadas ao texto como forma de pré-processamento. As utilizadas por esse trabalho, por apresentarem melhores resultados nas bases de dados utilizadas, são conhecidas como: tokenização,

capitalização, remoção de tags HTML, palavras de parada (*stop words*) e lematização.

Uma das etapas fundamentais do pré-processamento consiste na transformação de um documento em *tokens*. Um documento pode ser separado em partes menores dele, como sentenças ou palavras, descartando caracteres de pontuação, gerando-se assim *tokens* (MANNING *et al.*, 2009, tradução nossa). Através desse processo, estamos preparando o documento para a posterior extração de características, que será explicado na subseção a seguir.

Abaixo temos um exemplo de como funciona a tokenização. São dadas uma sentença e seus respectivos *tokens*.

Sentença	This is an example of tokenization.					
Tokens	This	is	an	example	of	tokenization

Tabela 1 - Exemplo de tokenização

Fonte: Própria (2018)

Dessa forma, fica evidente que cada palavra é transformada em um *token* (no caso da tokenização de sentenças). Nesse tipo de pré-processamento é possível evitar que pontuações sejam transformadas em *tokens*, visto que elas não agregam nenhum valor importante para nossos modelos.

Capitalização coloca todas as palavras de um documento em letras minúsculas (ou maiúsculas) a fim de que o mesmo vocábulo não seja colocado em *tokens* diferentes. “[...] isso é uma boa idéia: permitirá que instâncias de Automóveis no início de uma sentença correspondam a uma consulta de automóvel.” (MANNING *et al.*, 2009, tradução nossa). Evitando que sejam criados *tokens* diferentes da mesma palavra, estamos melhorando a qualidade do que será passado posteriormente para os algoritmos, gerando resultados melhores.

A remoção de tags HTML é feita no intuito de excluir tais entidades, pois elas não nos dizem nada de relevante ao texto. “Isso inclui extrair texto significativo de fontes de dados, como dados HTML, que consiste em tags HTML desnecessárias



[...].” (SARKAR, 2016). Dessa forma estamos retirando dados com ruído, ou seja, estamos removendo dados sem sentido.

Já a lematização, consiste em colocar a palavra em sua forma base, ou seja, ignorando o tempo verbal, palavras no plural e seu gênero. É o processo de realizar análises sobre a palavra, tanto léxica quanto morfológica, a fim de que possamos obter a sua forma base ou como é escrita no dicionário (MANNING *et al.*, 2009, tradução nossa). Utilizar a lematização é uma boa estratégia, pois assim é possível unificar todas as palavras que estão escritas diversas formas diferentes, fazendo com que assim tenhamos um *token* que representa todas essas variações do mesmo vocábulo.

Palavras de parada ou stop words são palavras que podem ser consideradas de pouco significado, ou seja, que não possui muita relevância em um texto. São palavras que geralmente não ajudam em uma busca e dessa forma podem ser excluídas (MANNING *et al.*, 2009, tradução nossa). Sendo assim, palavras de parada também podem ser consideradas como dados de ruído.

A figura abaixo mostra um pequeno conjunto de palavras de parada em inglês. Pode-se verificar realmente que estas não agregam nenhum grande significado ao texto.

a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

Figura 2 - Exemplo de palavras de parada

Fonte: (MANNING *ET AL.*, 2009)

Está claro que essas palavras não são palavras “fortes”, ou seja, que nos informam sobre o assunto principal de uma sentença. Geralmente sites de buscas descartam palavras de parada por esse motivo. No geral são preposições, conjunções, determinantes e alguns verbos de ligação.

## 2.4.2. EXTRAÇÃO DE CARACTERÍSTICAS DE TEXTOS

Para trabalhar com algoritmos de aprendizado de máquina utilizando textos, é necessário transformá-los em representações numéricas. Dessa forma, afirma-se detalhadamente:

A decisão sobre as características certas é parte integrante de um projeto de aprendizado de máquina bem sucedido. Embora as redes neurais profundas aliviem muita da necessidade de engenharia de características, um bom conjunto de características principais ainda precisa ser definido. Isto é especialmente verdadeiro para dados de linguagem, que vem na forma de uma sequência de símbolos discretos. Essa sequência precisa ser convertida de alguma forma em um vetor numérico, de uma maneira não óbvia. (GOLDBERG, 2017, tradução nossa).

De acordo com a citação acima, é esperado que os algoritmos de aprendizado de máquina não aceitem dados textuais, uma vez que dadas duas sentenças, ambas provavelmente terão tamanhos variáveis. Ao se trabalhar com números, fica mais fácil de utilizar vetores de tamanho fixo, além de que números são mais precisos.

O processo para utilizarmos sequências de textos nos algoritmos de aprendizado de máquina possui o nome de extração de características. “O mapeamento de dados textuais para vetores de valores reais é chamado de extração de características ou representação de características, e é feito por uma função de características.” (GOLDBERG, 2017). A extração de características é realizada após a etapa de pré-processamento, e é um passo muito importante, pois através de uma métrica bem definida (seja pela quantidade de vezes que uma palavra ocorreu ou de acordo com sua frequência) estamos obtendo valores reais que serão utilizados como dados de entrada. Logo, é preciso determinar qual função de características será utilizada a fim de que possamos transformar *tokens* em números para alimentar nossos modelos.

Um dos tipos de extração de características é feito pelo algoritmo *Bag of Words (BoW)*, que determina a quantidade de vezes que uma palavra aparece em cada documento. “Nesta abordagem, olhamos para o histograma das palavras dentro do texto, ou seja, considerando cada contagem de palavras como uma característica.” (GOLDBERG, 2017, tradução nossa). Em cada documento, toda

ocorrência do mesmo *token* será contada, ou seja, se por exemplo a palavra ‘computação’ aparece três vezes, seu *token* será mapeado para esse número, fazendo com que agrupemos palavras repetidas e dessa forma palavras que aparecem muitas vezes são um forte indicativo de um determinado tema.

A seguir, temos um exemplo de uma matriz Documento-Termo. Cada documento (linha) possui computado a quantidade de vezes que uma palavra (coluna) ocorre.

	I	love	computer	science	and	hate	history
Document 1	2	1	1	1	1	1	1
Document 2	1	1					1
Document 3	1	1	1	1			

Tabela 2 - Matriz Documento-Termo (*Bag of Words*)

Fonte: Própria (2018)

Outro tipo de extração de características se dá através do algoritmo *Tf-idf* (*Term frequency-inverse document frequency*), onde *term frequency* diz respeito à frequência na qual uma palavra aparece em um determinado documento. “Uma medida de quão importante uma palavra pode ser é o *term frequency* (tf), com que frequência uma palavra ocorre em um documento.” (ROBINSON; SILGE, 2017). Já o termo *inverse document frequency* diz respeito à frequência com que essa mesma palavra aparece em todos os outros documentos do *corpus*, indicando se a mesma deve ser considerada uma palavra de assinatura de um determinado documento. “Outra abordagem é examinar a *inverse document frequency* (idf) de um termo, o que diminui o peso das palavras comumente usadas e aumenta o peso das palavras que não são muito usadas em uma coleção de documentos.” (ROBINSON; SILGE, 2017). Apesar de a primeira vista parecer uma operação complexa, *Tf-idf* traz informações mais valiosas do que o algoritmo *BoW*, pois uma palavra pode aparecer muitas vezes em um documento. Porém, quanto mais vezes esse mesmo vocábulo aparecer em outros documentos, ele vai se tornando menos importante e provendo menos informações valiosas a um documento específico.

Abaixo, temos explicitada a equação do *Tf-idf*, onde o peso de uma palavra **x** em um documento **y** é dado pela frequência na qual **x** aparece em **y**, multiplicado

pelo logaritmo da divisão do número total **N** de documentos pelo número de documentos para qual **x** ocorre.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

Dessa forma, fica claro que para uma palavra ser importante em um determinado documento, ela deve ocorrer com mais frequência (primeiro termo) e deve ocorrer com menos frequência em todos os outros documentos, fazendo com que seu logaritmo se torne alto (segundo termo).

Temos também a seguir outra matriz Documento-Termo, porém dessa vez utilizando *Tf-idf*. Pode-se notar que as palavras mais importantes para cada documento estão destacadas em negrito.

	<b>I</b>	<b>love</b>	<b>computer</b>	<b>science</b>	<b>and</b>	<b>hate</b>	<b>history</b>
<b>Document 1</b>	0	0	0,18	0,18	0,48	0,48	0,18
<b>Document 2</b>	0	0					0,18
<b>Document 3</b>	0	0	0,18	0,18			

Tabela 3 - Matriz Documento-Termo (*Tf-idf*)

Fonte: Própria (2018)

Nota-se que quanto mais palavras em comum os documentos possuem, mais o peso das palavras diminui. Pesos altos são um forte indicativo do assunto principal do mesmo.

Outra forma de extração de características chama-se *Word Embeddings*, que representa uma palavra como sendo um vetor. É possível afirmar que:

O problema que as *Word embeddings* tentam resolver é como inserir palavras no espaço vetorial de baixa dimensão, de modo que as palavras semanticamente próximas fiquem próximas neste espaço, e palavras diferentes estejam distantes umas das outras. (GRIGOREV, 2017, tradução nossa).

A importância do *Word Embeddings* se dá ao fato de que é possível associar palavras, aproximando-as e revelando semânticas similares. Dessa forma, palavras que ocorrem próximas em um determinado texto, são próximas no vetor espacial.

## 2.5. CLASSIFICAÇÃO NAIVE BAYES

O classificador *Naive Bayes* é um algoritmo probabilístico empregado para resolver o problema da classificação. Seus modelos se encaixam perfeitamente quando dispomos de uma base de dados com várias dimensões, por se tratar de um algoritmo um tanto quanto simples, comparados com outros disponíveis, além de ser rápido e possuir uma quantidade baixa de parâmetros com os quais podemos mexer (VANDERPLAS, 2017).

Na equação abaixo, temos o teorema de Bayes. Ele consiste em calcular a probabilidade de um evento  $y$  (features) acontecer dado que  $x$  aconteceu ( $L$ ).

$$P(L | features) = \frac{P(features | L)P(L)}{P(features)}$$

O algoritmo *Naive Bayes* tem como pressuposto este teorema, que trabalha com probabilidade condicional. “Na classificação Bayesiana, estamos interessados em achar a probabilidade de uma *label* dada algumas características observadas, o que nós podemos escrever como  $P(L | features)$ .” (VANDERPLAS, 2017, tradução nossa). No problema da classificação binária, como é o caso do problema deste trabalho para identificação de *Fake News*, é necessário decidir entre duas *labels* ('FAKE'/'TRUE'); então, é preciso calcular a probabilidade a posteriori, ou seja, a classe mais provável é aquela classe com maior probabilidade. “Se estivermos tentando decidir entre dois rótulos - vamos chamá-los de  $L1$  e  $L2$  -, então, uma maneira de tomar essa decisão é calcular a proporção das probabilidades posteriores para cada rótulo.” (VANDERPLAS, 2017, tradução nossa).

Sendo assim, resta apenas o último passo que consiste para cada *label*, achar o valor de  $P(L | features)$ . Isso é feito através do chamado modelo *generativo*. “[...] especifica o processo aleatório hipotético que gera os dados. Especificar esse modelo generativo para cada rótulo é a parte principal do treinamento desse classificador bayesiano.” (VANDERPLAS, 2017, tradução nossa).

Existem dois diferentes tipos de algoritmos *Naive Bayes*, onde os mesmos possuem visões diferentes para o modelo generativo. “Diferentes tipos de

classificadores Naive Bayes baseiam-se em diferentes suposições *naive* sobre os dados [...].” (VANDERPLAS, 2017, tradução nossa). Neste trabalho veremos apenas o classificador *Multinomial Naive Bayes*, que foi o utilizado para o estudo do problema.

## 2.6. CLASSIFICAÇÃO SUPPORT VECTOR MACHINES KERNEL LINEAR

*Support Vector Machines* consiste em um classificador discriminativo para o aprendizado supervisionado, no qual o objetivo é encontrar o melhor hiperplano para dividir suas classes. “Um classificador discriminativo linear tentaria desenhar uma linha reta separando os dois conjuntos de dados e, assim, criar um modelo para classificação.” (VANDERPLAS, 2017, tradução nossa).

Abaixo, temos uma figura na qual é possível desenhar três hiperplanos. De todas estas maneiras conseguimos separar os conjuntos já existentes, cada um de um lado do hiperplano (os círculos vermelhos de um lado e os amarelos de outro).

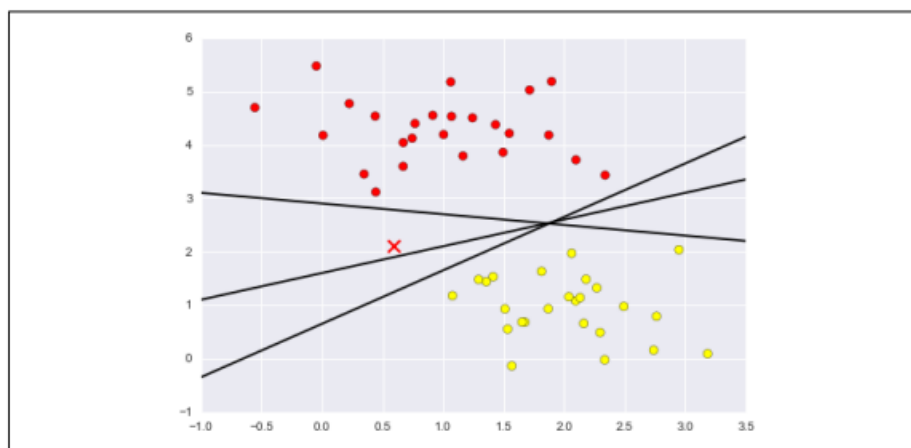


Figura 3 - Hiperplanos que dividem com sucesso o classificador SVM

Fonte: (VANDERPLAS, 2017)

No entanto, ao inserir um novo dado (representado por um 'x'), que pode ser tanto um círculo vermelho quanto um amarelo, não fica claro qual seria o melhor hiperplano. É preciso então encontrar uma forma de maximizar a sua margem. Definindo a margem máxima, ou seja, a maior distância do hiperplano para os pontos mais próximo do mesmo, em ambas as classes, encontramos a melhor reta possível (VANDERPLAS, 2017).

A seguir temos a representação da margem. É possível ver que ela usa como auxiliar, os pontos mais próximos do hiperplano.

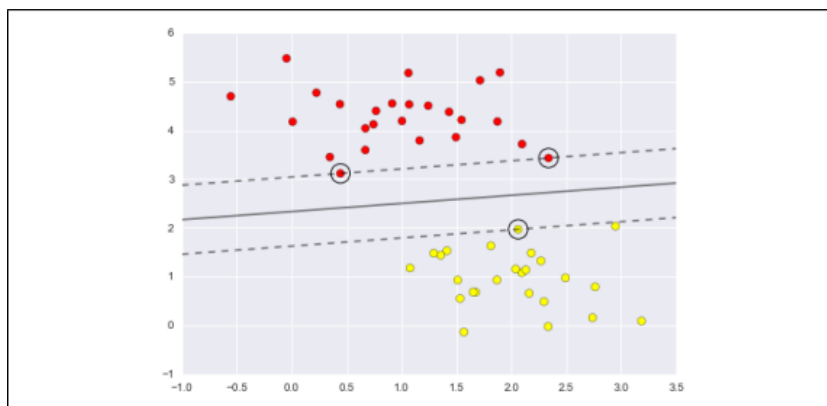


Figura 4 - Hiperplano ótimo - SVM

Fonte: (VANDERPLAS, 2017)

Tais pontos são chamados de vetores de suporte. “Os vetores de suporte definem totalmente o hiperplano de margem máxima ou a solução SVM, o que justifica o nome do algoritmo.” (MOHRI *et al.*, 2012, tradução nossa).

É importante lembrar que nessas figuras, a dimensão do hiperplano é dois, visto que o número das características de entrada também é dois, o que mostra uma relação entre os dois atributos. A dimensão do hiperplano aumenta de maneira igual de acordo com o número de *features*, ou seja, uma relação  $x = y$ , criando por exemplo uma reta no caso de duas *features* e um plano bidimensional no caso de três *features*.

## 2.7. DEEP LEARNING

Logo a seguir, é possível ver o universo onde o Deep Learning (*Deep Learning*) está inserido, sendo conhecido como uma subárea de Aprendizado de Máquina, que por sua vez é uma subárea de Inteligência Artificial.

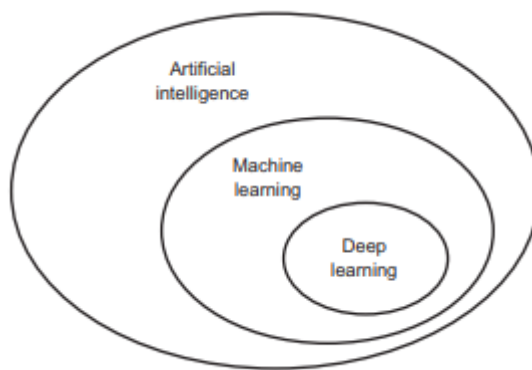


Figura 5 – Contexto no qual Deep Learning está inserido

Fonte: (CHOLLET, 2018)

Com isso, nota-se que Deep Learning é uma área do vasto campo de Aprendizado de Máquina. Apesar de ser uma subárea de *Aprendizado de Máquina*, a forma de aprender no Deep Learning é diferente, pois os dados de entrada passam por diversas camadas (CHOLLET, 2018).

Em Deep Learning os modelos, diferentemente dos modelos dos métodos tradicionais de Aprendizado de Máquina, são redes neurais. Esse tipo de modelo foi criado inspirado nas redes neurais do corpo humano. A respeito do Deep Learning:

No Deep Learning, essas representações em camadas são (quase sempre) aprendidas por meio de modelos denominados redes neurais, estruturados em camadas literais empilhadas umas sobre as outras. O termo rede neural é uma referência à neurobiologia, mas embora alguns dos conceitos centrais da aprendizagem profunda tenham sido desenvolvidos em parte pela inspiração de nossa compreensão do cérebro, os modelos de aprendizagem profunda não são modelos do cérebro. (CHOLLET, 2018).

De acordo com a citação acima, as redes neurais artificiais foram então inspiradas (e não copiadas) a partir do cérebro humano, que vão adquirindo conhecimento através da experiência, e a partir disso gerou-se um modelo matemático, possuindo diversas camadas no caso do Deep Learning.

Basicamente, cada camada armazena os dados que vem da camada imediatamente anterior, multiplicando tais dados pelos pesos, com o objetivo final de encontrar a classificação correta. Nesse sentido, pode-se afirmar:

Em termos técnicos, diríamos que a transformação implementada por uma camada é parametrizada por seus pesos [...]. (Os pesos também são às vezes chamados de parâmetros de uma camada.) Nesse contexto,



aprender significa encontrar um conjunto de valores para os pesos de todas as camadas em uma rede, de modo que a rede mapeie corretamente entradas de exemplo para seus destinos associados. (CHOLLET, 2018).

De acordo com o que foi explicado acima, a tarefa principal de uma rede neural artificial é encontrar os pesos. Os pesos fazem com que a rede se comporte de maneira adequada, sendo este a maior dificuldade do processo de uma rede neural artificial.

Encontrar os pesos corretos que resultem na saída esperada é uma tarefa repetitiva. Até que se encontre um conjunto de valores que se adaptem ao que se espera na saída, é necessário o auxílio da função de perda. A função de perda, ao obter os resultados gerados na saída de uma rede neural, compara o valor com o valor esperado e nos retorna um *score* de perda que nos diz o quão bem o modelo foi treinado (CHOLLET, 2018). Tal função é de extrema importância, pois ela é fundamental no processo para que a rede neural artificial encontre os valores de peso corretos e assim consiga uma predição satisfatória.

Caso o conjunto de pesos não mostre resultados razoáveis, os mesmos sofrerão mudanças a fim de diminuir o *score*. “Esse ajuste é o trabalho do otimizador, que implementa o que é chamado de algoritmo Backpropagation [...]” (CHOLLET, 2018). Tal fato evidencia o quão mais desenvolvida é o Deep Learning em relação aos métodos tradicionais de Aprendizado de Máquina.

## 2.8. REDE NEURAL DE CONVOLUÇÃO

Rede Neural de Convolução (ou *Convolutional Neural Network – CNN*) é considerada uma das principais categorias para se trabalhar na área de visão computacional, pois atua fortemente nas tarefas de reconhecimento e classificação de imagens. “[...] funcionam particularmente bem em problemas de visão computacional, devido à sua capacidade de operar de maneira convolucional, extraíndo características de *patches* de entradas locais [...]” (CHOLLET, 2018, tradução nossa).

Para esse tipo de rede, diferentemente da ordem global da informação, a ordem local é de extrema importância na tarefa de classificação. Em relação a *CNN*:

A rede neural de convolução é desenvolvida para identificar preditores locais em uma grande estrutura, e para combiná-los para produzir uma representação vetorial de tamanho fixo da estrutura, capturando os aspectos locais que são mais informativos para a tarefa de previsão disponível. (GOLDBERG, 2017, tradução nossa).

De acordo com a citação acima, é possível combinar aspectos locais a fim de obtermos resultados mais consistentes. Dessa forma, estamos agregando mais informações no momento de realizar uma predição.

Da mesma maneira que a rede neural de convolução funciona tão bem para imagens, ela também pode funcionar para textos, porque também estamos interessados em encontrar padrões específicos, olhando a ordem local de uma sentença (por exemplo), onde dada uma combinação de palavras é possível ter um significado diferente de quando trabalhamos com palavras isoladas. Através da combinação de palavras, podemos encontrar um indício de qual classificação uma determinada sentença se encaixa, fazendo isso sem olhar para onde a mesma está inserida, utilizando as camadas de convolução e de *pooling* (GOLDBERG, 2017).

A figura abaixo mostra o processo da convolução, onde temos um Kernel (*convolution window* ou *sliding window*) de tamanho igual a cinco, de uma dimensão, junto com *pooling*, que é utilizado na rede neural de convolução.

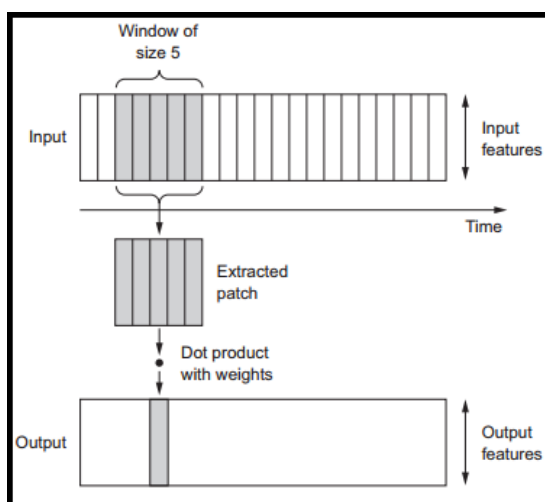


Figura 6 - Convolução 1D

Fonte: (CHOLLET, 2018)

Inicialmente temos um vetor de palavras que juntos formam uma sentença e utilizaremos todas elas em partes, através do tamanho do kernel (que no caso acima é igual a cinco). Com essa parte da sentença é realizado o produto escalar, que nos diz características importantes das palavras “capturadas”. Vale lembrar que é possível ter mais de um filtro, sendo que nesse caso teremos um vetor do tamanho do mesmo, cada um contendo seu produto escalar, fazendo parte do vetor de saída apenas aquele com maior valor (no caso de mais de um filtro), com o auxílio da camada de *pooling*. Dessa forma o processo é repetido enquanto houver palavras a serem consideradas (o filtro vai deslizar para a próxima porção de palavras, o que chamamos de *sliding window*). De maneira precisa, temos que:

A idéia principal por trás da arquitetura de convolução e *pooling* para tarefas de linguagens é aplicar uma função não-linear (aprendido) através de cada janela deslizando de k-palavra sobre a sentença. Esta função (também chamada de “filtro”) transforma a janela de k-palavras em um valor escalar[...]. Então, uma operação “pooling” é usada para combinar os vetores resultantes de diferentes janelas em um único vetor l-dimensional, pegando o máximo ou o valor médio observado em cada uma das dimensões l sobre diferentes janelas. (GOLDBERG, 2017).

Dessa maneira, através da convolução, é possível combinar palavras próximas, ou seja, de maneira local em relação ao texto. Através da camada de pooling, junta-se os melhores resultados obtidos. A importância da CNN se dá justamente pelo fato de que com esse modelo pode-se juntar os melhores resultados obtidos durante o processo, contribuindo para a classificação.

A equação a seguir traz o conceito de Max-pooling, um dos tipos de *pooling* utilizados após a convolução:

$$c_{[j]} = \max_{1 \leq i \leq m} p_{i[j]} \quad \forall j \in [1, l]$$

Essa é a fórmula mais utilizada, consistindo em, para cada vetor produzido pela operação de convolução, irá fazer parte do vetor de saída de tamanho l, aquele que possuir o maior valor escalar. (GOLDBERG, 2017).

## 2.9. REDE NEURAL DE RECORRÊNCIA

A Rede Neural de Recorrência (*Recurrency Neural Network* ou *RNN*) possui uma estrutura de loop, ou seja, consegue a cada instante de tempo tomar uma decisão baseada não somente na nova entrada, mas também baseado no que foi aprendido de entradas anteriores, sendo esse esquema chamado de conexões recorrentes. Sobre esse termo, é possível afirmar:

As conexões recorrentes fornecem uma rede recorrente com visibilidade não apenas da amostra de dados atual que foi fornecida, mas também do estado oculto anterior. Uma rede recorrente com um loop de feedback pode ser visualizada como várias cópias de uma rede neural, com a saída de uma que servindo como uma entrada para a próxima. Ao contrário das redes neurais tradicionais, as redes recorrentes usam sua compreensão de eventos passados para processar o vetor de entrada em vez de começar do zero todas as vezes. (NVIDIA, 2018).

Podendo olhar tudo o que foi processado anteriormente, temos a vantagem de poder tomar melhores decisões. Em um texto, é possível ter uma noção melhor do contexto em qual uma determinada palavra está inserida, e dessa forma podemos verificar que *RNN* leva vantagem em relação a *CNN* visto que esta última olha apenas localmente.

A figura a seguir explicita o funcionamento de uma conexão recorrente. Temos que cada **input** é uma entrada de dados e cada **output** é uma saída, e cada **State** representa a saída do nó imediatamente anterior.

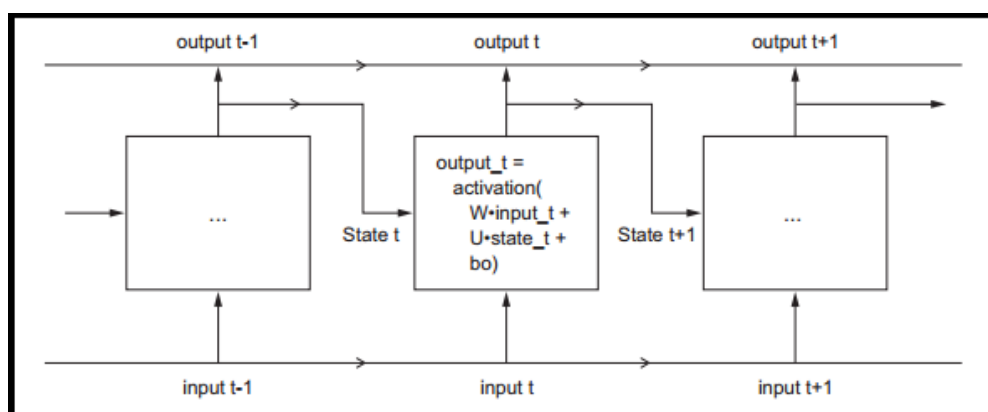


Figura 7 - Rede Neural de Recorrência

Fonte: (CHOLLET, 2018)

Dessa forma, a saída para cada nó leva em consideração não apenas a nova entrada de dados, como também o valor de **State**, comprovando que nesse tipo de rede levam-se em consideração tudo o que foi visto anteriormente.

## 2.10. LONG SHORT-TERM MEMORY

Uma das instâncias da *RNN* é chamada de *LSTM* (*Long Short-Term Memory*), na qual se tem um aumento na memória, aumentando a capacidade do modelo de armazenar dados anteriores em uma sequência. De forma concisa, define-se:

A arquitetura LSTM divide o vetor estado  $s_t$  em duas metades, onde uma metade é tratada como “células de memória” e a outra como memória de trabalho. As células de memória são designadas para preservar a memória, e também o erro dos gradientes, através do tempo, e são controladas através dos componentes de disparos diferenciáveis – funções matemáticas suaves que simulam portas lógicas. [...] uma porta é usada para decidir quanto da nova entrada deve ser escrita na célula de memória, e quanto do conteúdo atual da célula de memória deve ser esquecido. (GOLDBERG, 2017).

*LSTM* possui a vantagem de definir o que pode ser enviado para a porta lógica do esquecimento, definindo assim o que não tem mais utilidade no treinamento. Com uma memória maior, é possível definir ainda melhor o contexto no qual algo está inserido.

De maneira bem estruturada, as portas lógicas possuem funções bem definidas, na qual dado uma nova entrada, ela só poderá ser armazenada na célula de memória se o modelo definir que este dado possui relevância e então pode entrar na porta de entrada, definir que um dado pode ser encaminhado para a porta de saída e contribuir para a saída ou então enviar dados que não são mais necessários, ou seja, o contexto não precisa mais ser utilizado, para a porta de esquecimento (GOLDBERG, 2017).

## 2.11. TRABALHOS RELACIONADOS

Este capítulo diz respeito aos trabalhos relacionados que contribuíram para o desenvolvimento deste trabalho de conclusão de curso. Colaboraram para o mesmo o artigo *Text Classification Using Aprendizado de Máquina Techniques* (IKONOMAKIS; KOTSIANTIS; TAMPAKAS, 2005), *Convolutional Neural Networks*

*for Sentence Classification* (KIM, 2014) e *LSTM Recurrent Neural Networks for Short Text and Sentiment Classification* (NOWAK; TASPINAR; SCHERER, 2017).

O primeiro artigo diz respeito ao problema da classificação de gênero, dado um determinado texto. O problema é resolvido utilizando aprendizagem de máquina supervisionado. O artigo também chama atenção para as etapas do processo de classificação de texto e menciona o problema do número de características, que podem atingir números elevados, afetando no processo de classificação pelos algoritmos de *Aprendizado de Máquina*. Por fim, são discutidas alguns algoritmos que podem ser utilizados, apresentando desempenhos diferentes como *Naive Bayes*, *Support Vector Machines*, *Decision Tree* e *kNN*, além de técnicas para avaliar a performance dos algoritmos sobre este problema.

No segundo artigo são feitos experimentos sobre as *Convolutional Neural Networks*, modificando seus parâmetros e utilizando *word2vec*, a fim de explicitar o ótimo desempenho alcançado por esta rede sobre o problema de classificação de sentenças, demonstrando que uma *CNN* com uma camada de convolução alcança uma ótima acurácia. No experimento também foram realizadas mudanças na construção da *CNN*, apresentando quatro variações.

Já para o último artigo citado, foi realizado um experimento com o objetivo de classificar textos utilizando *LSTM*, *Bidirectional LSTM* e *GRU*, sendo as duas últimas variações de *LSTM*. Foram utilizados na etapa de treinamento as bases de dados *Spambase*, *Farm Advertisement* e *Amazon book reviews*, com o intuito de demonstrar como o desempenho destas redes neurais são melhores do que alguns algoritmos, como *AdaBoost* e *bag-of-words*.

### 3. METODOLOGIA

Este trabalho seguiu uma abordagem experimental e exploratória no intuito de avaliar quais abordagens (Aprendizado de Máquina ou Deep Learning) possuem melhor desempenho na classificação de *fake news*, dados tamanhos de *datasets* diferentes. A pesquisa foi desenvolvida a partir de fontes secundárias (livros, relatórios, artigos e conteúdos de sites), a fim de obter conceitos e aprender técnicas de Aprendizados de Máquina e Profundo. Os resultados serão traduzidos de forma quantitativa, os quais serão apresentados através de gráficos e tabelas.

#### 3.1. CONFIGURAÇÃO DAS PLATAFORMAS DE EXECUÇÃO

A parte de desenvolvimento (programação) do trabalho foi realizada utilizando a linguagem Python, através do Jupyter Notebook, que consiste em uma aplicação web *open-source* interativa na qual é possível programar de forma organizada e mostrar partes do código de forma eficiente. Também foi utilizada a biblioteca SpaCy, desenvolvida para processamento avançado de linguagem natural. É importante ressaltar as configurações dos computadores nos quais a parte da programação foi realizada, para que se possam contextualizar as limitações impostas pela mesma, que serão discutidas no capítulo de conclusão. O computador utilizado para o treinamento dos métodos tradicionais de Aprendizado de máquina possui as seguintes configurações:

- Sistema operacional Windows 10 – 64 bits.
- Processador Intel Core i5 – 6200U 2.3 GHz.
- Memória RAM de 8GB.
- HD de 1TB

Para o treinamento dos modelos baseados em Deep Learning foi utilizada uma máquina virtual da suíte de computação em nuvem Google Cloud Platform. As configurações são dadas por:

- Sistema operacional Windows Server 2016 – 64 bits.

- GPU NVIDIA Tesla P100
- Memória RAM de 30GB
- 10 vCPUs
- HD de 50GB

### 3.2. BASES DE DADOS

Na tabela abaixo se pode verificar a estrutura das bases de dados que foram utilizadas. Foram produzidas três versões que foram classificadas como: pequena, média e grande.

Base	Amostras	Amostras de Treinamento	Amostras de Teste	Notícias Falsas	Notícias Reais
Pequena	1.000	670	330	308	692
Média	10.000	6700	3300	3171	6829
Grande	100.000	67000	33000	32051	67949

Tabela 4 - Dados dos datasets

Fonte: Própria (2018)

Cada tabela possui um número de amostras (1.000, 10.000 e 100.000), que foram separadas em amostras de 67% dos dados para treinamento e 33% para teste. A razão da criação de três bases de dados se deu pelo fato de querer verificar quais modelos possuem o melhor desempenho em cada um dos *datasets*.

A seguir, temos a estrutura contida em todas as bases de dados. Cada uma possui cinco colunas que indicam a identificação, o site, o conteúdo e o título da notícia.

id	domain	type	content	title
6583	1074	christianpost.com	REAL	(Photo: REUTERS/Jason Lee) A cross is seen beh... Chinese Gov't Fines Pastors Over \$1M, Demandin...
6039	1249	beforeitsnews.com	FAKE	Healthy to Eat, Unhealthy to Grow: Strawberrie... Healthy to Eat, Unhealthy to Grow: Strawberrie...
4001	3052	christianpost.com	REAL	(Photo: Reuters/Jayanta Shaw) School children ... Thousands of Indian Christians Protest After P...
7137	5201	beforeitsnews.com	FAKE	Headline: Bitcoin & Blockchain Searches Exceed... Jordan Peterson: Everything you do matters. Tr...
9245	6548	beforeitsnews.com	FAKE	Looming Catastrophe Hanging Over Our Heads – G... Looming Catastrophe Hanging Over Our Heads – G...

Tabela 5 – Estrutura da base de dados utilizada

Fonte: Própria (2018)



Tais bases são versões modificadas do *dataset* criado por Maciej Szpakowski, ex-aluno da Universidade de *Southampton*.<sup>1</sup> A base de dados é *open source*, e até o momento da conclusão deste trabalho possui 9.408.908 de notícias, classificadas em 7 tipos/tags, incluindo *Fake News/fake* e *Credible/reliable*. Como estamos lidando com um problema de classificação binária, ou seja, queremos saber se uma notícia é falsa ou não, para as três versões criadas, somente notícias classificadas com estas duas *tags* citadas anteriormente serão levadas em conta, porém renomeadas como ‘*FAKE*’ e ‘*REAL*’.

### 3.3. EXECUÇÃO DA METODOLOGIA

Foram implementados modelos de classificação tradicionais (*Multinomial Naive Bayes* e *Support Vector Machine*) e de Deep Learning (*LSTM* e *CNN*). Depois foi feita a realização de um experimento que nos permitisse verificar quais foram os parâmetros que eram necessários modificar para obtermos a melhor acurácia em cada modelo, verificar qual o melhor modelo (ainda em termos de acurácia) tradicional e baseado em *deep learning* e qual é o melhor modelo dentre todos. Todos esses experimentos foram feitos baseados na mesma base de dados, contendo tamanhos diferentes.

Para preparar o *dataset* para ser utilizado no processo de classificação, foi realizado no mesmo o pré-processamento de seus dados. As principais técnicas que foram utilizadas são a *tokenização*, *lowercasing*, *lematização*, remoção de *stop words*, pontuação e *tokens* que não consistem de caracteres alfabéticos e de ASCII.

Tanto o classificador *Multinomial Naive Bayes* quanto o classificador *Support Vector Machine* foram construídos com dois extratores de características: *bag-of-words* (*CountVectorizer*) e *tf-idf* (*TfidfVectorizer*). Vale lembrar, que nesses dois modelos, foi decidido que termos que aparecem em menos de 2% e mais de 60% do documento (pois esta configuração gerou melhores resultados) são descartados. Já para a redes neurais *CNN* e *LSTM*, foi utilizado word embeddings, porém primeiro foi

---

<sup>1</sup> <https://github.com/several27/FakeNewsCorpus>

preciso transformar textos em sequências. É importante lembrar que foi feito preenchimento da sequência de palavras com o número zero (***padding*** = a soma de todas as sequências/(tamanho dos dados de treinamento + tamanho dos dados de teste)) para que seja possível igualar o tamanho das sequências.

Com os resultados obtidos sobre os *datasets* após o pré-processamento de dados e posterior extração de características foi possível realizar uma análise comparativa do melhor valor (que gera a melhor acurácia) de acordo com alguns parâmetros para cada algoritmo. Tais parâmetros foram escolhidos após diversos testes, apresentando maiores influências no resultado final. Para o algoritmo que utiliza o classificador *Multinomial Naive Bayes*, foram realizados testes sobre os parâmetros '**alpha**' (de **0.01** a **1**), que é um valor adicionado a cada probabilidade de uma *feature* a fim de evitar probabilidades iguais a zero, e '**fit\_prior**' (**True** ou **False**), que diz se queremos que o modelo aprenda com probabilidades anteriores de uma classe, e assim foi possível visualizar qual a combinação desses dois valores retorna a melhor acurácia possível. Da mesma forma, para o algoritmo que utiliza o classificador *Support Vector Machine*, o parâmetro '**C**' (de **0.01** a **1**), que é a constante de regularização que nos diz o quanto queremos evitar erros de classificação, também possibilitou verificar qual valor retorna a melhor acurácia. Os resultados obtidos foram colocados em um gráfico de linhas simples, evidenciando a acurácia obtida de acordo com os valores dos parâmetros.

Para os modelos de *deep learning*, a técnica *early stopping* (com condição de parada caso tenham dez ocorrências de erros no conjunto de validação que sejam maiores que o menor valor desse erro), que evita problemas de underfitting ou overfitting, foi usada para que possamos ter um limite do número de épocas para o treinamento. Também são gerados arquivos que guardam os valores dos pesos, sempre que há uma melhora no mesmo de acordo com o *epochs* atual, para que possamos gravar as informações da maior acurácia do modelo e assim gerar a matriz de confusão. Após esta etapa, foram gerados dois gráficos de linhas simples, um contendo a acurácia do conjunto de treino e de validação para cada época, e outro contendo o *score* da função de perda dos mesmos conjuntos, também para cada época.

A figura a seguir descreve a estrutura utilizada neste trabalho para o modelo *CNN*. Embora esteja mostrando a versão com *dataset* pequeno, a mesma estrutura de camadas se aplica às outras opções de bases de dados.

Layer (type)	Output Shape	Param #
embedding_26 (Embedding)	(None, 495, 58)	1660656
conv1d_26 (Conv1D)	(None, 495, 64)	18624
global_max_pooling1d_25 (Glo	(None, 64)	0
dropout_26 (Dropout)	(None, 64)	0
dense_51 (Dense)	(None, 10)	650
dense_52 (Dense)	(None, 1)	11
Total params: 1,679,941		
Trainable params: 1,679,941		
Non-trainable params: 0		

Figura 8 - Estrutura de camadas para CNN com dataset pequeno

Fonte: Própria (2018)

Importante lembrar que algumas opções de parâmetros foram testadas, e as mesmas serão explicitadas no capítulo seguinte (Análise experimental).

Abaixo, está explicitada também a estrutura utilizada para o modelo *LSTM*. Assim como para o modelo *CNN*, a figura mostra a versão com o *dataset* pequeno, porém a mesma estrutura foi aplicada para as bases de dados de tamanho médio e grande.

Layer (type)	Output Shape	Param #
embedding_109 (Embedding)	(None, 495, 58)	1660656
cu_dnnlstm_98 (CuDNNLSTM)	(None, 64)	31744
dropout_114 (Dropout)	(None, 64)	0
dense_132 (Dense)	(None, 10)	650
dense_133 (Dense)	(None, 1)	11
Total params: 1,693,061		
Trainable params: 1,693,061		
Non-trainable params: 0		

Figura 9 - Estrutura de camadas para LSTM com dataset pequeno

Fonte: Própria (2018)

Foram testados alguns parâmetros para esse modelo, que também estão explicitados no capítulo a seguir.

A partir da obtenção dos melhores parâmetros, os classificadores e classes de redes neurais foram treinados, gerando uma matriz de confusão, que evidencia o número de notícias que foram classificadas corretamente e erroneamente não só como '*FAKE*', mas também como '*REAL*'. Assim, foi possível verificar qual classificador de métodos tradicionais possui a melhor precisão, utilizando *bag-of-words* ou *tf-idf*, e em seguida foi possível verificar qual dos dois classificadores tradicionais, junto com o extrator de características, possui a melhor acurácia, de acordo com o tamanho da base de dados. De forma parecida, para as redes neurais artificiais, também foi possível verificar qual das duas resultam na melhor acurácia, dado um *dataset*. Finalizando, foi possível comparar qual método é o melhor de todos em cada base de dados.

## 4. ANÁLISE EXPERIMENTAL

### 4.1. ABORDAGEM TRADICIONAL

Para a abordagem tradicional foram utilizados os classificadores *Multinomial Naive Bayes* e *Support Vector Machines*. Ambos foram treinados considerando dois extratores de características: *Bag-of-Words* e *Tf-idf*. Os modelos foram treinados variando os parâmetros '*alpha*' (de **0.01** até **1**) que é o parâmetro de suavização aditiva e '*fit\_prior*' (**True/False**) que decide se o modelo deve aprender probabilidades da classe anterior ou não (em caso negativo, uma probabilidade anterior uniforme será usada) para o caso do *Multinomial Naive Bayes* e o parâmetro '*C*' (corresponde ao parâmetro de penalidade do termo de erro) para o caso do *Support Vector Machines*.

#### 4.1.1. MULTINOMIAL NAIVE BAYES COM BAG-OF-WORDS

Utilizando agora *Multinomial Naive Bayes* com *Bag-of-Words*, para o *dataset* pequeno, vê-se que com o parâmetro *fit\_prior* = *True*, conseguimos o melhor *score* do modelo, sendo este igual a 0.85 para *alpha* = 0,02 (lembrando mais uma vez que está sendo considerado o primeiro a ser achado).

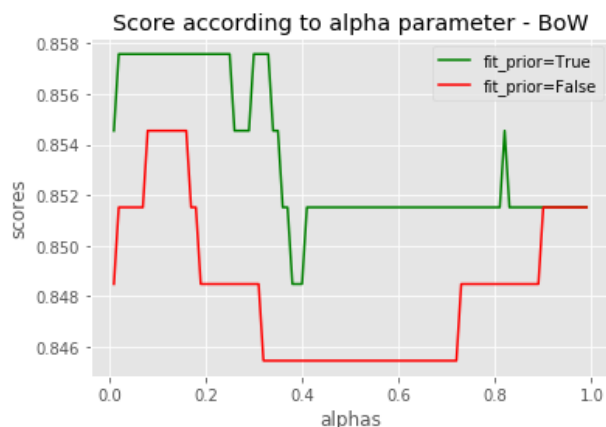


Gráfico 1 - *MNB* com *BoW* e 1.000 dados: *alphas* x *scores*

Fonte: Própria (2018)

Em alguns casos, *fit\_prior* = *False* gera valores melhores maiores de *alpha* (melhor valor nesse caso é alcançado utilizando o valor de 0,02 e gerando *score* =

0,858) do que em alguns instantes utilizando a variável booleana como verdadeira. Porém nunca ultrapassa o melhor *score*. Nota-se que para o *dataset* com mil dados, o modelo possui áreas de instabilidade e de estabilidade em todo o gráfico.

Na figura abaixo temos a representação da matriz de confusão, que nos mostra um erro de classificação de 14% e acurácia de 86%. Nota-se que estes valores são piores do que no extrator de característica *Tf-idf*, utilizando a mesma base (mil dados) e o mesmo modelo.

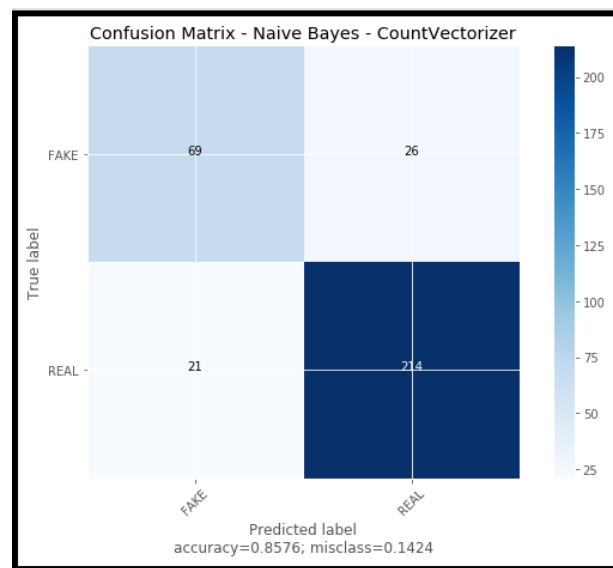


Figura 10 - MNB com BoW e 1.000 dados: matriz de confusão

Fonte: Própria (2018)

Dos 95 dados classificados como falsos 69 estão corretos e 26 não estão. Já para os dados classificados como verdadeiros 214 de fatos são verdadeiros e 21 não o são, de 235 dados.

Para o *dataset* de tamanho médio, *fit\_prior = True* ainda gera os melhores resultados. O melhor *score* (0,86182) foi obtido utilizando este valor *booleano* com *alpha = 0,57*.

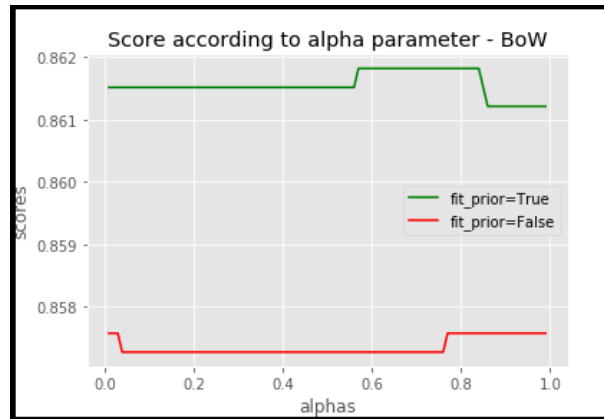


Gráfico 2 - MNB com BoW e 10.000 dados: *alphas* x *scores*

Fonte: Própria (2018)

Assim como no caso em que utilizamos *Tf-idf*, para essa base de dados, nota-se que o gráfico começa a se estabilizar, gerando valores quase constantes. Vale observar que pior parâmetro (*fit\_prior = False*), o melhor resultado se deu em *alpha* = 0,01 gerando *score* = 0,85758.

Abaixo, temos a matriz de confusão para a base de dados com dez mil notícias. A porcentagem do erro de classificação chegou a 14% e a acurácia ficou em 86%, assim como no *dataset* pequeno (ao fazer o arredondamento dos valores).

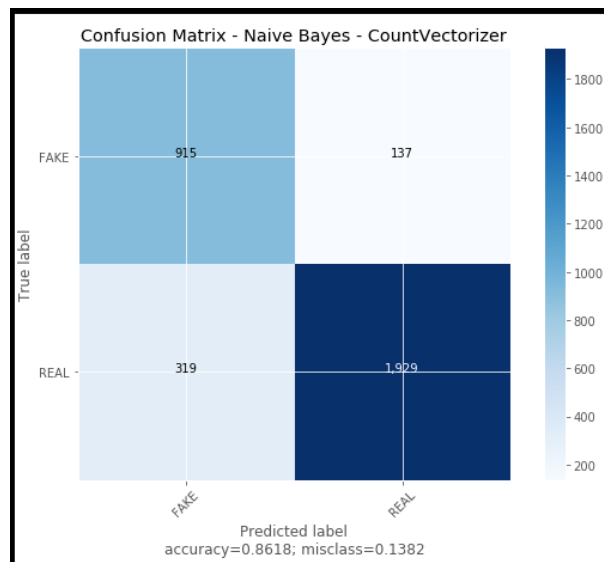


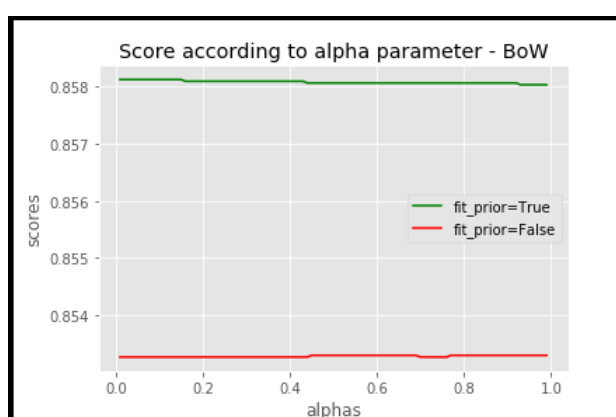
Figura 11 - MNB com BoW e 10.000 dados: matriz de confusão

Fonte: Própria (2018)

Das 1.057 notícias classificadas como falsas, 915 foram classificadas corretamente e 137 não foram. Da mesma forma, para as notícias verdadeira, 1.929

foram classificadas corretamente e 319 não o foram, de 2.248 notícias verdadeiras. Verifica-se que o *dataset* pequeno e o médio basicamente possuem o mesmo desempenho.

Em relação à base de dados de cem mil notícias, os valores de acurácia são quase constantes para a faixa de *alphas*, assim como no caso do *dataset* grande utilizando *Tf-idf*. O parâmetro *fit\_prior = True*, assim como nas outras bases, gerou o melhor *score* (0,85812, com *alpha* = 0,01). O gráfico a seguir deixa claro este fato.



**Gráfico 3 - MNB com BoW e 100.000 dados: *alphas* x *scores***

**Fonte: Própria (2018)**

Fica claro mais uma vez que com bases de dados grandes, os *scores* vão se tornando constantes. Apesar da diferença de valores gerados entre *fit\_prior* sendo *True* ou *False* (neste último o melhor resultado foi em *alpha* = 0,01 gerando *score* = 0,85327) ser pequena, ainda sim é válido escolher a combinação de parâmetros que gera o melhor resultado.

Na figura abaixo, mostra a última matriz do caso *Multinomial Naive Bayes* com *Bag-of-words*. O erro de classificação ficou em 14% e a acurácia em 85%, assim como nos outros dois casos. Isso demonstra que a performance deste modelo com *BoW* é o mesmo independente do tamanho da base de dados (para este trabalho).



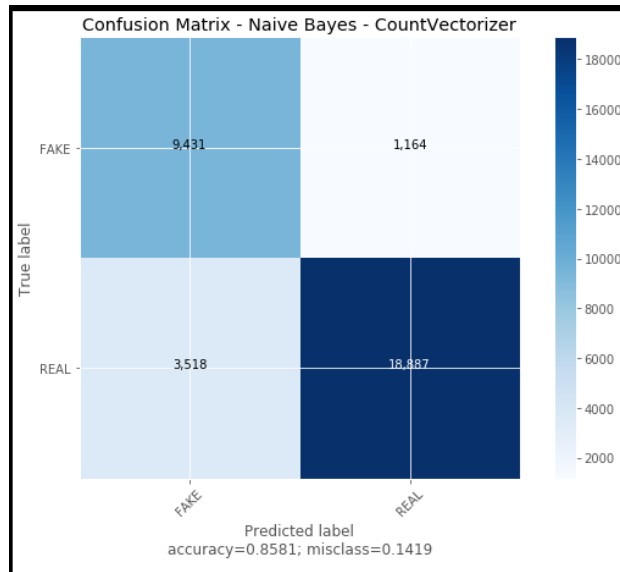


Figura 12 - MNB com BoW e 100.000 dados: matriz de confusão

Fonte: Própria (2018)

Das 10.595 notícias falsas, 9.431 foram classificadas corretamente e 1.164 não o foram. Já para as notícias verdadeiras, das 22.405 notícias verdadeiras, 18.887 estão corretas e 3.518 não estão.

#### 4.1.2. MULTINOMIAL NAIVE BAYES COM TF-IDF

Treinando para mil dados (*dataset* pequeno), nota-se que a acurácia é maior quando utilizamos *fit\_prior = False*. Os resultados podem ser visualizados no gráfico a seguir.



Gráfico 4 - MNB com Tf-idf e 1.000 dados: *alphas x scores*

Fonte: Própria (2018)

Utilizando o parâmetro *fit\_prior = True*, percebe-se que a partir de um determinado momento (quando *alpha = 0,11*, que foi o primeiro encontrado, produzindo *score = 0,86*), quanto mais aumentarmos o valor de *alpha*, menor será o *score* produzido. Dessa maneira, a melhor alternativa é utilizar *fit\_prior = False*, pois se pode visualizar que os resultados obtidos são superiores ao primeiro caso, atingindo seu auge em *alpha = 0,86* com *score 0,89*.

Na figura abaixo é mostrada a matriz de confusão, que nos revela os resultados previstos para o conjunto de teste. A acurácia obtida foi de 89%, como já foi dito anteriormente. Essa matriz também nos retorna a porcentagem de classificações erradas, sendo de 11% neste exemplo.

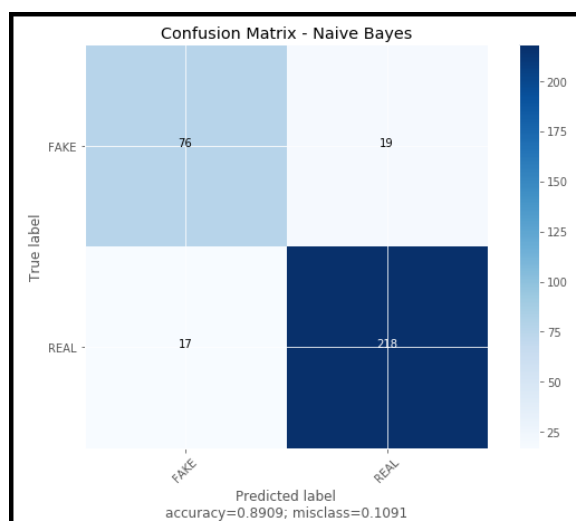


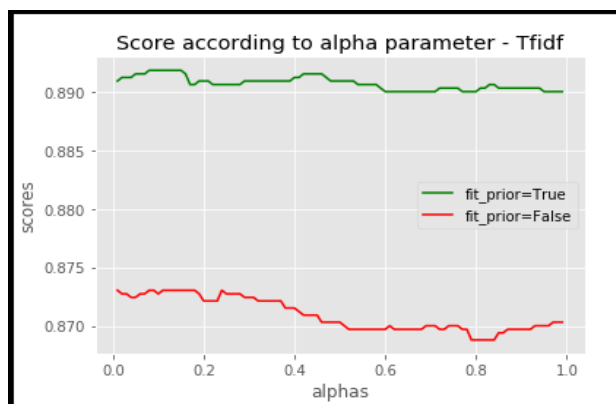
Figura 13 - MNB com Tf-idf e 1.000 dados: matriz de confusão

Fonte: Própria (2018)

Na matriz de confusão, é possível visualizar com clareza o número de notícias que foram classificadas de maneira correta e de maneira errada. Verifica-se que das 95 notícias falsas, 76 foram classificadas corretamente e por outro lado 19 foram classificadas de maneira errada. Da mesma forma, das 235 notícias verdadeiras, 218 tiveram sua classificação acertada e apenas 17 foram erradas.

Quando treinamos para mil dados (*dataset* médio), verificou-se que utilizar o parâmetro *fit\_prior = True* agora é a melhor opção. Diferentemente do caso para mil dados, vemos que com dez mil a variação do *score* se deu de maneira mais suave,

ou seja, a variação de no valor de *alpha* não afeta a estabilidade do score gerado. O gráfico a seguir explicita essa situação.

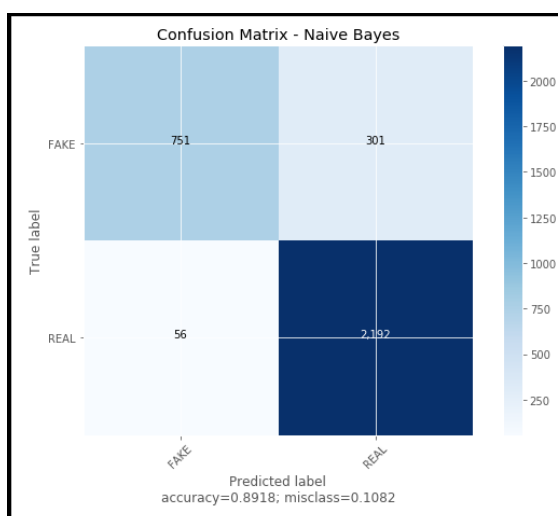


**Gráfico 5 - MNB com Tf-idf e 10.000 dados: *alphas* x *scores***

**Fonte: Própria (2018)**

Vê-se que a melhor combinação de parâmetros com a base de dados com dez mil notícias se dá utilizando *alpha* = 0,08, obtendo um score de 0,89. Com *fit\_prior* = *False*, o score mais alto (0,873) também é obtido com *alpha* sendo 0,08. Fica mais evidente ainda que o parâmetro *fit\_prior* é um dos mais decisivos no modelo *Multinomial Naive Bayes*.

Na matriz de confusão gerada, no total 11% de todas as notícias foram classificadas de maneira errada e 89% de maneira correta. Pode-se verificar tal fato na figura abaixo.



**Figura 14 - MNB com Tf-idf e 10.000 dados: matriz de confusão**

**Fonte: Própria (2018)**

De 1.052 notícias falsas, 751 foram classificadas corretamente e 301 o foram. Já para as ditas como verdadeiras, de 2.248 notícias 2.192 estão com a classe correta e 56 não estão. Aumentar o tamanho da base de dados em 100 vezes não teve nenhum tipo de alteração (positiva ou negativa) no resultado final. Dessa forma, para modelos de tamanhos pequenos e médios (de acordo com o que foi estipulado para este trabalho), o desempenho do modelo *Multinomial Naive Bayes* é o mesmo.

Com o maior *dataset* trabalho (cem mil dados), o parâmetro *fit\_prior = True* produziu melhores resultados. O gráfico a seguir nos revela tal fato.

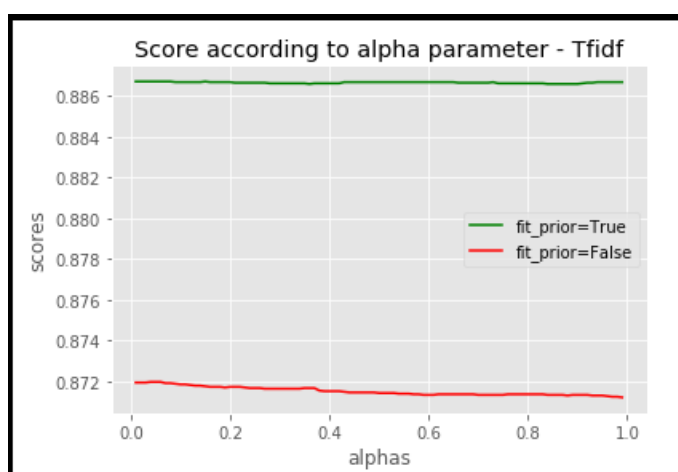


Gráfico 6 - MNB com Tf-idf e 100.000 dados: *alphas x scores*

Fonte: Própria (2018)

Com cem mil dados, observa-se que a estabilidade dos *scores* é ainda maior do que no caso anterior. Conclui-se então que quanto mais dados, maior será a possibilidade do parâmetro *alpha* não influenciar no resultado. A melhor combinação de parâmetros para o *dataset* grande se deu utilizando *alpha* como 0,01 e *fit\_prior = True*, gerando *score* = 0,89. Com *fit\_prior = False*, o melhor *score* foi 0,87 também com *alpha* = 0.01.

A última matriz de confusão para o modelo *Multinomial Naive Bayes* com *Tf-idf* evidencia mais uma vez o erro de classificação igual a 11% e a acurácia de 89%. A figura abaixo explicita os dados.

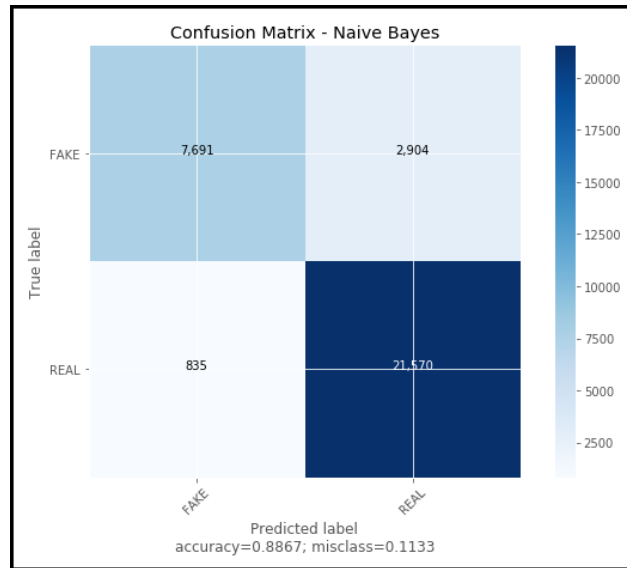


Figura 15 - MNB com Tf-idf e 100.000 dados: matriz de confusão

Fonte: Própria (2018)

Na base com cem mil notícias, 7.691 foram classificadas de maneira correta e 2.904 de maneira errada, em relação às notícias verdadeiras. Já para as falsas, de 22.405 notícias, 21.570 foram classificadas corretamente e 835 de maneira errada.

#### 4.1.3. SUPPORT VECTOR MACHINES COM BOW

Com o extrator de características *Bag-of-Words*, para o *dataset* com mil dados, é possível observar que dependendo do valor escolhido para o parâmetro C, o mesmo pode variar bastante ou permanecer constante. A partir de  $C = 0,55$ , verifica-se que o score se não altera mais (para o intervalo utilizado).

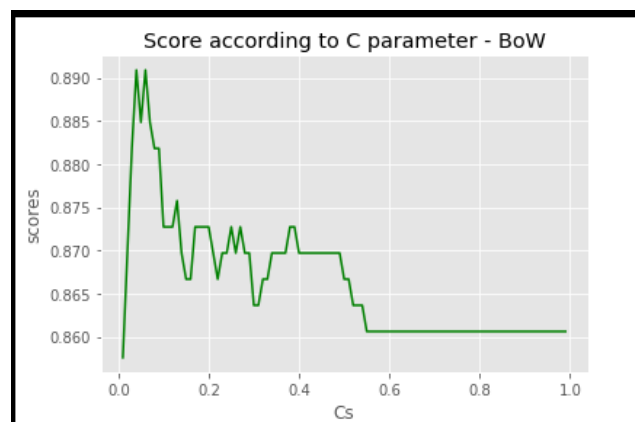


Gráfico 7 - SVM com BoW E 1.000 dados: Cs X scores

Fonte: Própria (2018)

Temos que a escolha que vai proporcionar é melhor acurácia, é em  $C = 0,04$  com  $score = 0,89091$ . Esse valor é um pouco menor do que utilizando *Tf-idf* para o *dataset* pequeno.

A seguir, na representação da matriz de confusão, possui a porcentagem do erro de classificação igual a 11% e a acurácia igual a 89%. Em comparação com o modelo utilizando *Tf-idf*, observa-se que este último citado é apenas um pouco superior, possuindo erro de classificação de 10% e acurácia de 90%.

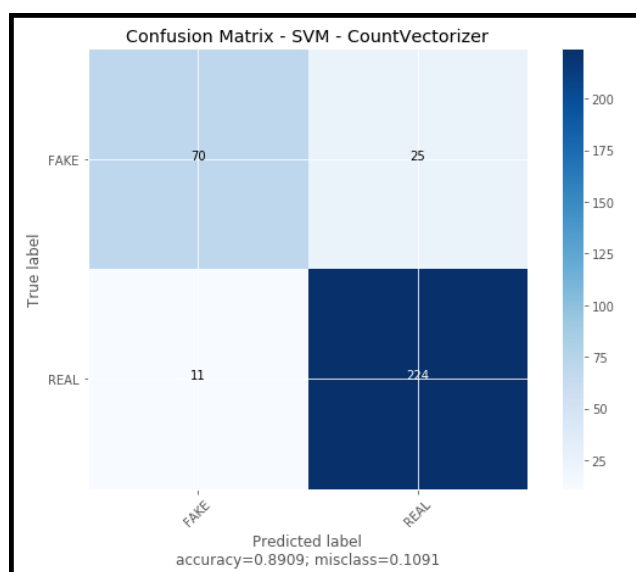
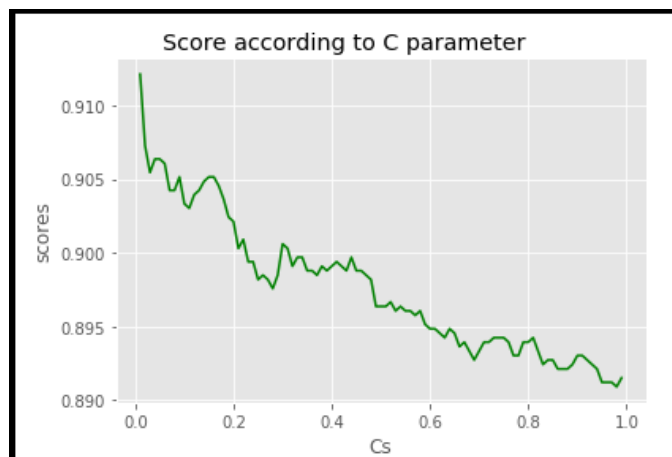


Figura 16 - SVM com BoW e 1.000 dados: matriz de confusão

Fonte: Própria (2018)

Dos 95 dados classificados como falsos, 70 deles foram previstos de maneira correta e 25 não o foram. Dos 235 dados classificados como verdadeiros, 224 de fato foram classificados corretamente e 11 não estão com sua *label* correta.

Para a base de dados de tamanho médio, observa-se que quanto mais aumentamos o valor do parâmetro  $C$ , mais a acurácia diminui. Seu melhor valor é atingido logo no início do intervalo.

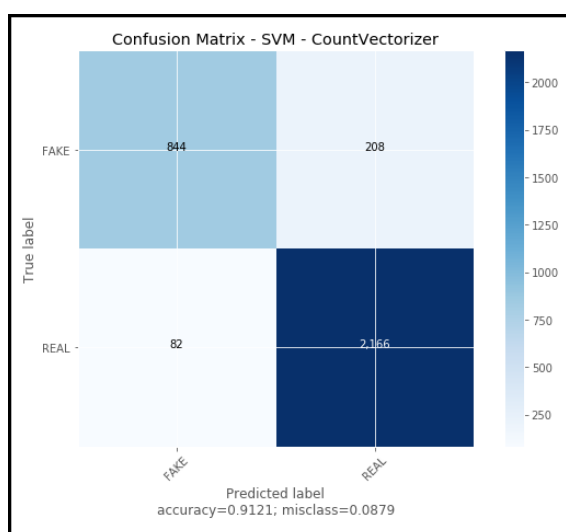


**Gráfico 8 - SVM com BOW e 10.000 dados: Cs x scores**

**Fonte: Própria (2018)**

Com uma acurácia de 0,91212, colocando o valor 0,01 no parâmetro C nos retorna a melhor acurácia do intervalo. Valor esse que confirma que *Support Vector Machine* com *BoW* possui melhor performance com o *dataset* médio do que com o pequeno, e que um valor muito pequeno de C retorna uma acurácia muito grande.

Na matriz de confusão, o erro de classificação é de 8% e acurácia de 91%. Esses valores são melhores do que para o *dataset* pequeno (13% e 86%), porém são piores se comparados com SVM utilizando *Tf-idf* e o mesmo tamanho de base (93% e 7%).

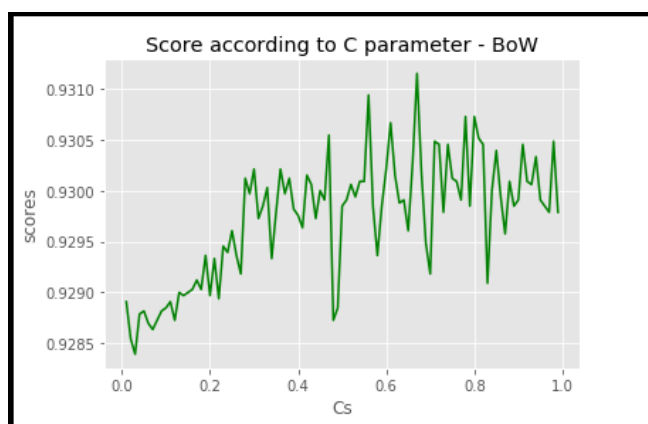


**Figura 17 - SVM com Bow e 10.000 dados: matriz de confusão**

**Fonte: Própria (2018)**

Das 1.052 notícias classificadas como falsas, de fato 844 estão classificadas corretamente e 208 não estão. Já para as notícias verdadeiras, de 2.248 notícias classificadas como verdadeiras, 2.166 realmente estão classificadas de maneira correta e 82 não estão. Essa abordagem é superior ao modelo *Multinomial Naive Bayes* com *Bag-of-Words*, que possui acurácia de 86% e erro de classificação de 13%.

Por fim, para o *dataset* contendo cem mil notícias, nota-se que o comportamento do gráfico é total instável, visto que a cada variação do parâmetro C, o *score* pode aumentar ou diminuir. O gráfico abaixo explicita tal situação.



**Gráfico 9 - SVM com BoW e 100.000 dados: Cs X scores**

**Fonte: Própria (2018)**

Apesar dessa instabilidade, o melhor parâmetro para esse intervalo de tempo foi dado em  $C = 0,67$  onde o *score* atingiu 0,93115. Logo, apesar deste comportamento, ainda sim o *dataset* grande retorna melhores resultados se comparado com as outras bases de dados para SVM com BoW.

A matriz de confusão nos mostra que o erro de classificação ficou em 7% e a acurácia em 93%. Isso deixa explícito que é o melhor caso deste modelo.



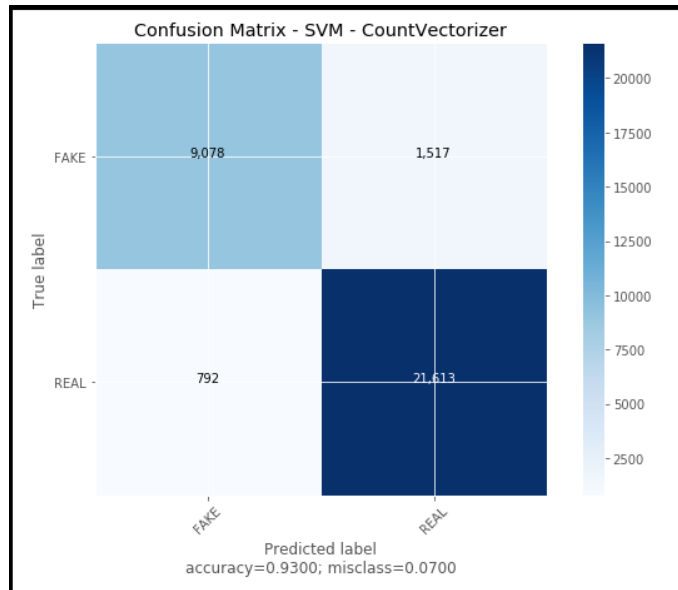


Figura 18 - SVM com BOW e 100.000 dados: Matriz de confusão

Fonte: Própria (2018)

Das 10.595 notícias falsas, 9.078 foram classificadas corretamente e 1.517 não o foram. Já para as verdadeiras, de 22.405 notícias, 21.613 estão com suas *labels* corretas e 792 não estão.

#### 4.1.4. SUPPORT MACHINE VECTORS COM TF-IDF

Falando agora sobre *Support Vector Machines*, e lembrando que foram testados uma faixa de valores do parâmetro C, para a base de dados pequena quanto maior o valor do parâmetro, maior é o *score* até que a partir de C = 0,61 os valores gerados começar a não variar muito. O gráfico a seguir mostra tal fato.

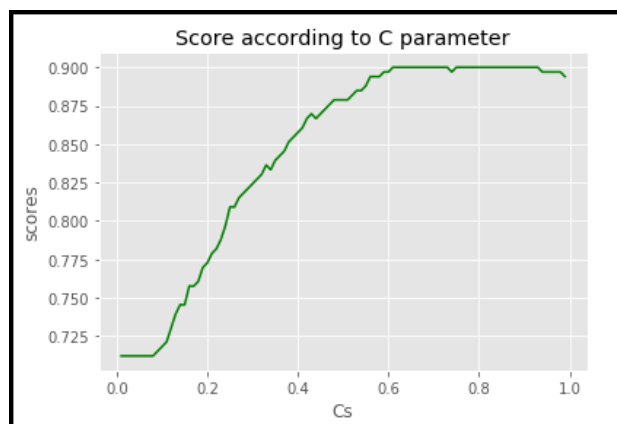


Gráfico 10 - SVM com *Tf-idf* e 1.000 dados: *Cs* x *scores*

Fonte: Própria (2018)

O melhor valor para este modelo com *Tf-idf* e utilizando o *dataset* pequeno é dado por 0,61 e gerando o *score* = 0,90. A importância do parâmetro C fica explicitada no gráfico, pois caso escolhêssemos um valor muito pequeno, a acurácia do modelo seria prejudicada.

A matriz de confusão mostra que o erro de classificação é de apenas 10% e a acurácia é de 90%. Abaixo, temos a representação desta matriz.

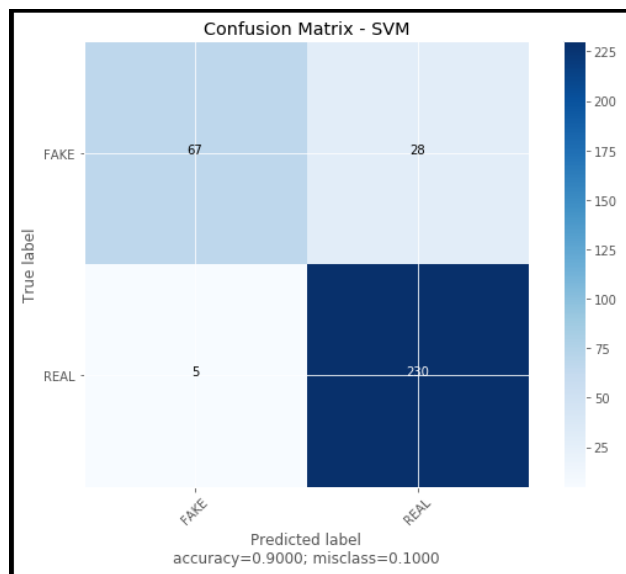
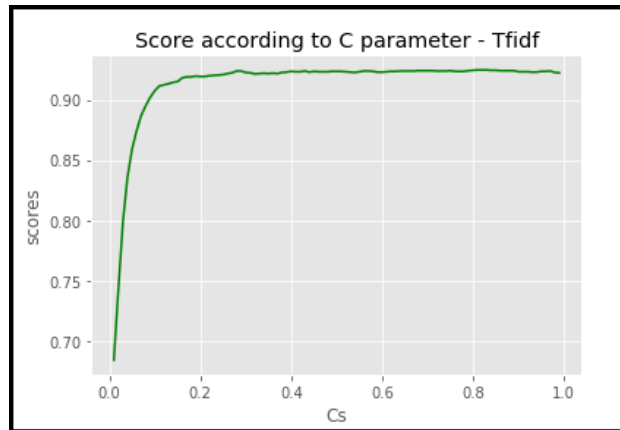


Figura 19 - SVM com *Tf-idf* e 1.000 dados: matriz de confusão

Fonte: Própria (2018)

É possível verificar que de 95 notícias falsas, 67 foram classificadas corretamente e 28 foram classificadas de maneira errada. Já para as notícias verdadeiras, 230 estão corretas e 5 estão erradas, de 235 notícias verdadeiras. Já é possível verificar que o desempenho deste modelo para base de dados pequena é melhor do que no caso em que foi utilizada do *Multinomial Naive Bayes* com *Tf-idf*.

Para a base de dados com dez mil notícias, percebe-se que o valor deixa de variar muito logo no início do intervalo, a ponto de quase se tornar constante. Para valores muito próximos de zero, a acurácia é bastante prejudicada.

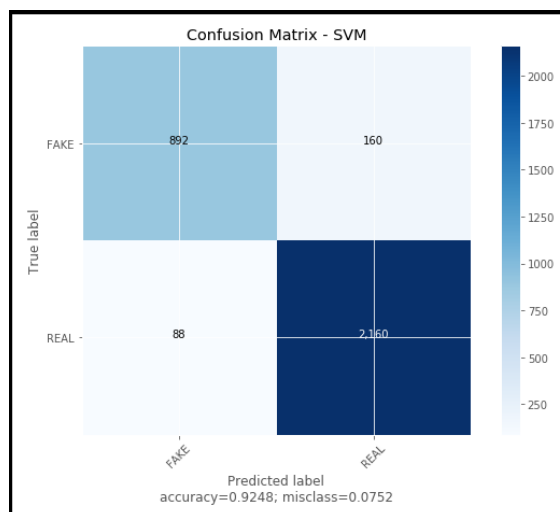


**Gráfico 11 - SVM com Tf-idf e 10.000 dados: Cs x scores**

**Fonte: Própria (2018)**

A melhor escolha de valor do parâmetro C no *dataset* médio é 0,81, onde *score* atinge 0,92485. Dessa maneira, nota-se que esse modelo possui melhor performance com a base de dados média do que com a de tamanho pequeno, aumentando sua acurácia em 2%. Ainda, sua acurácia também é melhor do que no modelo *Multinomial Naive Bayes* com *Tf-idf* (89%) e utilizando o mesmo *dataset* de tamanho médio.

A matriz de confusão a seguir demonstra que o erro de classificação (7%) diminuiu ainda mais do que no modelo com base de dados de mil notícias (10%). Dessa forma, temos mais uma maneira de visualizar que o desempenho com a base de dez mil notícias é melhor.

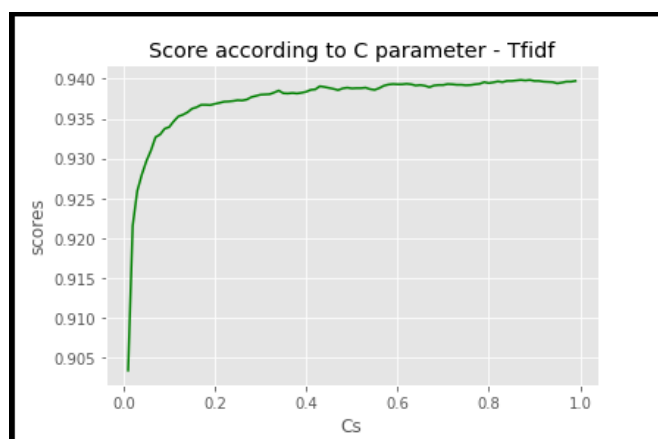


**Figura 20 - SVM com Tf-idf e 10.000 dados: matriz de confusão**

**Fonte: Própria (2018)**

Pode-se perceber, que de 1.052 notícias falsas, 892 foram classificadas corretamente e 160 não o foram. Já nas notícias verdadeiras, de 2.248 delas, 2.160 estão com a saída correta e 88 não estão.

Apesar de parecido com o gráfico para o *dataset médio*, no *dataset grande* que está explicitado a seguir, possui um valor de C que será uma acurácia superior. Esse valor é dado por 0,87 e gerando um *score* de 0,93985.



**Gráfico 12 - SVM com Tf-idf e 100.000 dados: Cs x scores**

**Fonte: Própria (2018)**

No entanto, o comportamento do gráfico ainda sim é bastante parecido. Conclui-se então que o modelo *Support Vector Machines* com *Tf-idf* utilizando o *dataset grande* possui a melhor performance (embora não tão superior se comparado com a base média) dentre as outras duas bases.

A seguir, a matriz de confusão mostra que o erro de classificação caiu mais uma vez e possui o valor de 6%. Isso demonstra que este modelo atingiu um valor muito satisfatório no processo de classificar uma notícia como sendo falsa ou não.

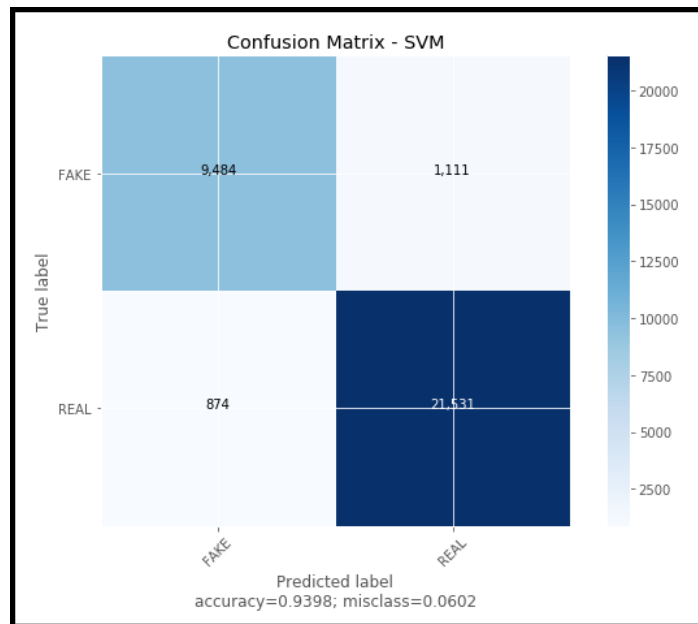


Figura 21 - SVM com Tf-idf e 100.000 dados: matriz de confusão

Fonte: Própria (2018)

Das 10.595 notícias falsas, 9.484 foram classificadas corretamente e 1.111 foram classificadas com a label errada ('*REAL*'). Já para as notícias verdadeiras, de 22.405, 21.531 foram classificadas com a label correta e 874 não o foram. Também vale dar destaque a acurácia de 94%.

## 4.2. ABORDAGEM DEEP LEARNING

Nos modelos de Deep Learning, a obtenção da melhor acurácia é feita através da procura do número de épocas (*epochs*), sendo utilizada a técnica early stopping, como explicado no capítulo de Metodologia, para evitar que ocorra underfitting (pode acontecer caso utilizemos poucas épocas) ou overfitting (pode acontecer caso utilizemos muitas épocas). É importante frisar que o número de *batch\_size* escolhido para os *datasets* pequeno, médio e grande são respectivamente: 32, 64 e 128. Esses números foram escolhidos após diversos testes a fim de encontrar o valor ideal, pois um *batch\_size* muito pequeno ou muito grande podem afetar no resultado final.

#### 4.2.1. CONVOLUTIONAL NEURAL NETWORK

Para o *dataset* pequeno é possível observar que o conjunto de validação atingiu uma acurácia aceitável, de maneira quase consistente, oscilando pouco. É possível observar que nos primeiros valores do parâmetro *epochs* (de zero a cinco), a acurácia atinge valores não satisfatórios.

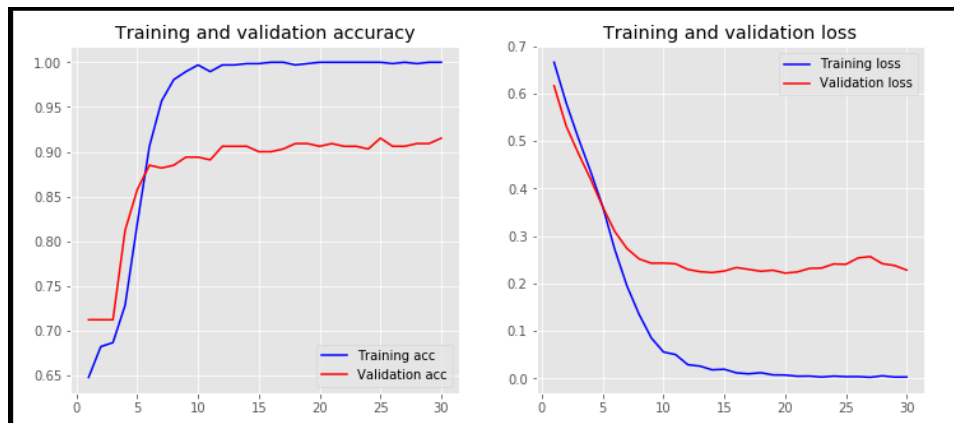


Gráfico 13 - CNN com 1.000 dados: gráficos de acurácia e perda

Fonte: Própria (2018)

É possível observar que o erro de validação é menor quando *epochs* = 22 gerando um valor de 0.2219. Após isso, a condição de parada não encontrou valores melhores de erro, sendo indicativo de um possível overfitting. É possível notar pelo gráfico que 25 é o melhor número de épocas para o *dataset* pequeno, atingindo acurácia de 91.52%.

Pela matriz de confusão abaixo, se pode observar que o erro de classificação para a base de dados contendo mil notícias é baixo (8%). A acurácia chega a 92%, sendo possível concluir que *CNN* pode ter um bom desempenho com bases de dados pequenas.

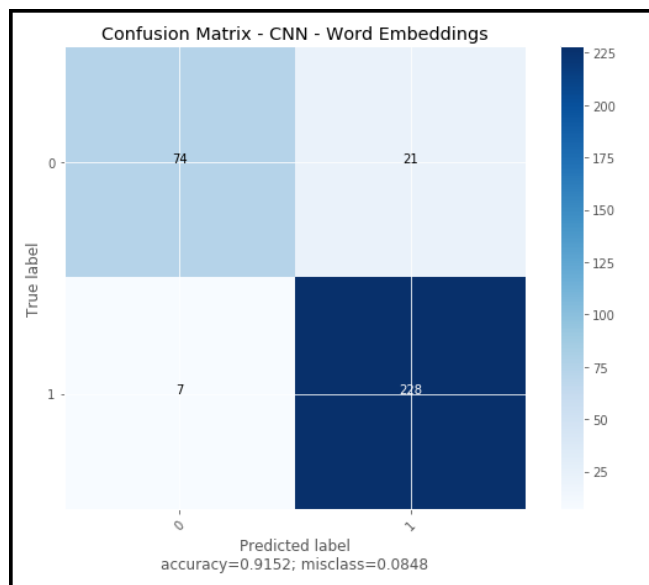


Figura 22 - CNN com 1.000 dados: matriz de confusão

Fonte: Própria (2018)

Pode-se ver pela figura que de 95 notícias falsas, 74 foram classificadas sem erros e 21 foram classificadas erroneamente. No caso das verdadeiras, de 235 notícias, 228 estão classificadas de maneira correta e apenas 7 não estão.

Para a base de dados contendo dez mil notícias, o conjunto de validação atinge uma acurácia ótima, sendo esta quase que constante ao longo do número de épocas. A partir de *epochs* = 2, sua acurácia passa dos 96%. O gráfico a seguir mostra tal situação.

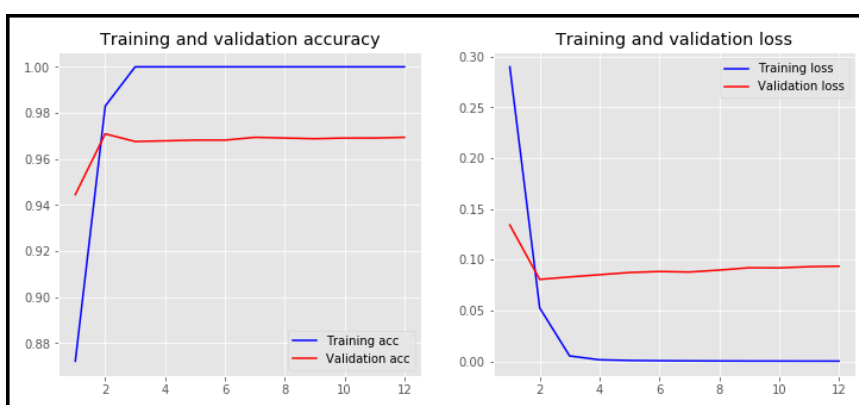


Gráfico 14 - CNN com 10.000 dados: gráficos de acurácia e perda

Fonte: Própria (2018)

Pode-se observar que a melhor acurácia é 97.09% que é atingida em *epochs* = 2. A partir desse valor, onde também encontramos o menor erro de validação, o mesmo não melhora e chega à condição de para indicando que pode ocorrer overfitting. Em relação ao *dataset* pequeno, o de tamanho médio possui melhor desempenho.

Na figura abaixo percebe-se que o erro de classificação é muito baixo, chegando a apenas 3%. Sua acurácia conforme indicado anteriormente é de 97%. Esses fatos evidenciam que para uma base de tamanho média, o problema de classificar se uma notícia é falsa ou não pode ser resolvido com *CNN*, trazendo ótimos resultados.

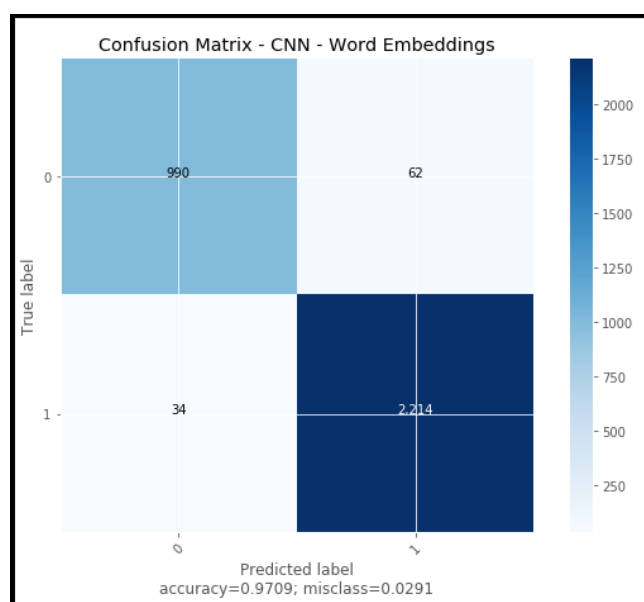


Figura 23 - CNN com 10.000 dados: matriz de confusão

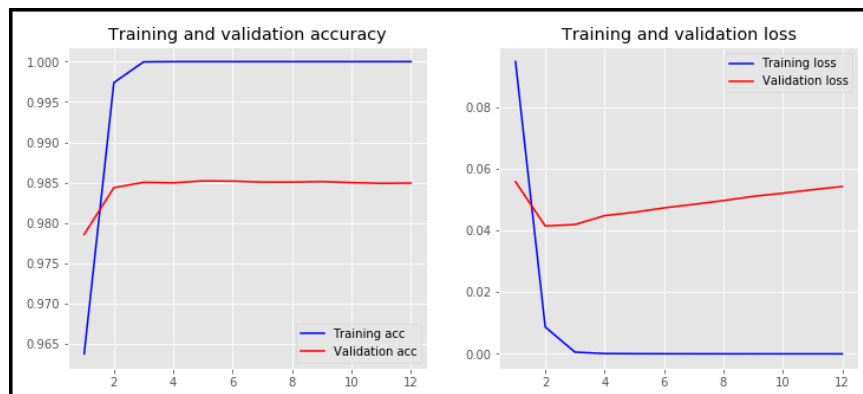
Fonte: Própria (2018)

Das 1.052 notícias falsas, 990 foram classificadas de maneira correta e 62 não o foram. Já para as notícias verdadeiras, 2.214 foram classificadas corretamente e 34 não o foram, de 2.248.

Para o *dataset* grande, em *epochs* = 5 temos a acurácia no conjunto de validação de 98.52%. Pode-se perceber que, assim como para a base de dados média, na base de dados grande o valor da acurácia não modifica muito, indicando



que quanto maior a *dataset*, mais comum é de se atingir valores constantes ou próximos de constantes.

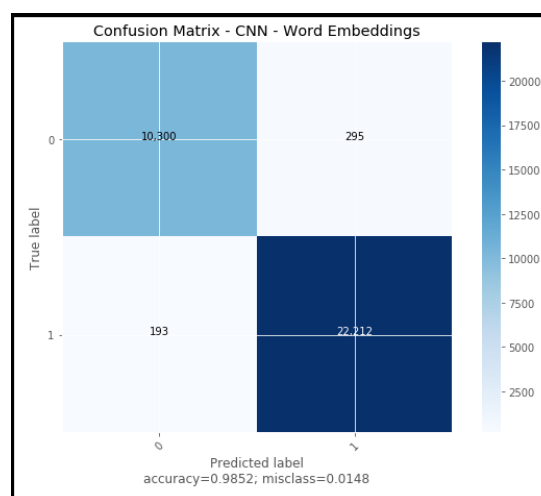


**Gráfico 15 - CNN com 100.000 dados: gráficos de acurácia e perda**

**Fonte: Própria (2018)**

*CNN* demonstrou ser um ótimo modelo para bases de dados grandes. Após *epochs* = 2, o erro de validação não melhora e começa a aumentar. Dessa forma, temos mais um indicativo de *overfitting*, fazendo com que a condição de parada atue.

Na matriz abaixo, verifica-se que o erro de classificação chegou a um valor muito baixo (1%), sendo este o melhor valor encontrado em todos os modelos. A acurácia, chegou a 99% (aproximadamente).



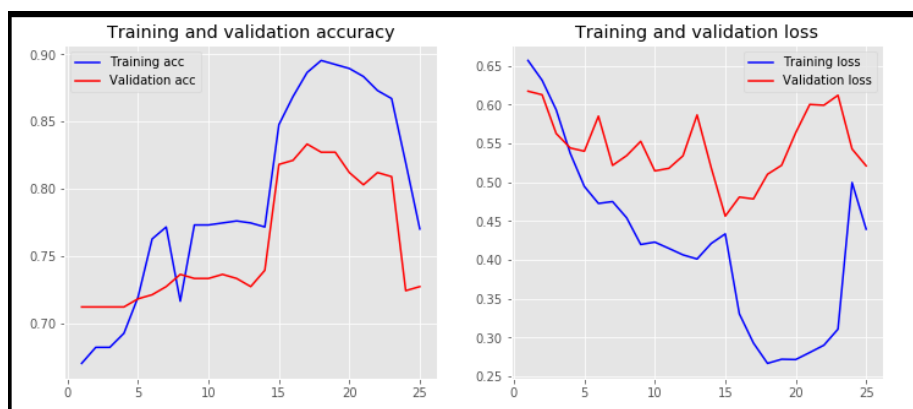
**Figura 24 - CNN com 100.000 dados: matriz de confusão**

**Fonte: Própria (2018)**

Pode-se notar que de 10.595 notícias falsas, 10.300 estão com suas *labels* corretas e 295 não estão. Já para as notícias verdadeiras, de 22.405 temos que 22.212 estão classificadas corretamente e 193 não estão.

#### 4.2.2. LONG SHORT-TERM MEMORY

Para o *dataset* contendo mil notícias, o modelo gerou um gráfico muito instável e com valores de acurácia ruins. Sua melhor acurácia é de apenas 83% em *epochs* = 16. O gráfico abaixo explicita a situação.



**Gráfico 16 - LSTM com 1.000 dados: gráficos de acurácia e perda**

**Fonte: Própria (2018)**

Assim como o gráfico de acurácia, o gráfico de erro também contém oscilações. Seu menor erro foi encontrado em *epochs* = 15. Numa condição assim, se torna difícil dizer que vai ocorrer *overfitting*, porém foram feitos testes com condições de paradas maiores e o comportamento do gráfico permanece com a mesma característica oscilatória.

A matriz de confusão a seguir reitera o fato de que este modelo não possui desempenho satisfatório para o *dataset* pequeno, com erro de classificação igual a 17%. Sua acurácia conforme explicitado anteriormente é de 83%.

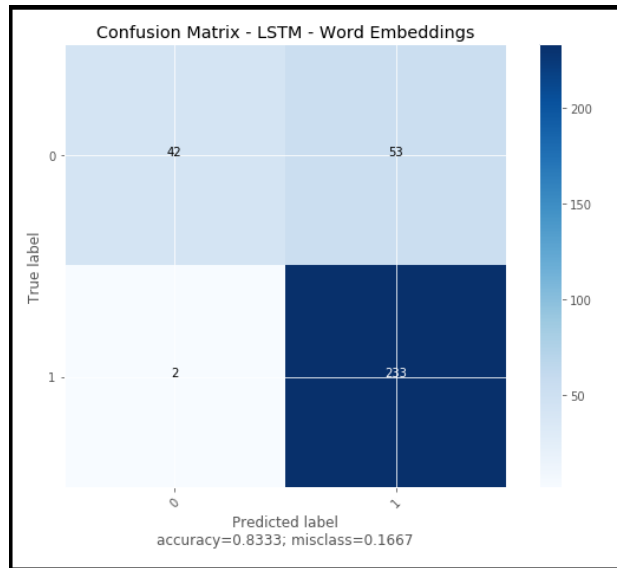


Figura 25 - LSTM com 1.000 dados: matriz de confusão

Fonte: Própria (2018)

A quantidade de notícias falsas classificadas de maneira incorreta superaram as que de fato estão com a *label* correta. De 95, temos que 53 foram classificadas erroneamente e 42 estão corretas. Já para as notícias verdadeiras, o cenário é diferente, pois a maioria teve sua classificação prevista corretamente, sendo 233 de um total de 235. Porém, isto não muda o fato do modelo ter um desempenho ruim.

Para a base de dados média, percebe-se que o gráfico começa a oscilar menos, em relação ao gráfico gerado utilizando a base de dados pequena. Apesar disso, os resultados ainda não estão muito satisfatórios.

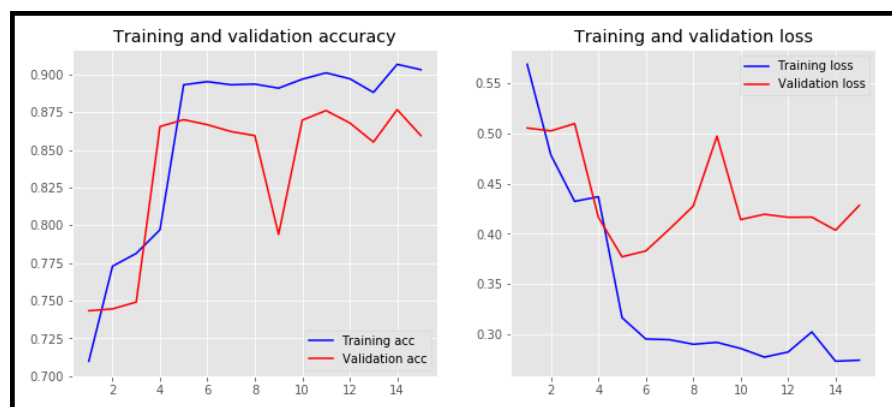


Gráfico 17 - LSTM com 10.000 dados: gráficos de acurácia e perda

Fonte: Própria (2018)

A maior acurácia foi atingida em *epochs* = 14 com 87,67%. Para o gráfico de erros, o menor valor de erro no conjunto de validação foi dado em *epochs* = 5, resultando em 0.3370.

A matriz de confusão abaixo nos mostra que o erro de classificação (16%) diminuiu um pouco em relação à base de dados pequena (17%). Isto reforça o fato de que utilizar o modelo com uma quantidade de dados razoável possui melhor desempenho com que usá-lo com base contendo poucos dados.

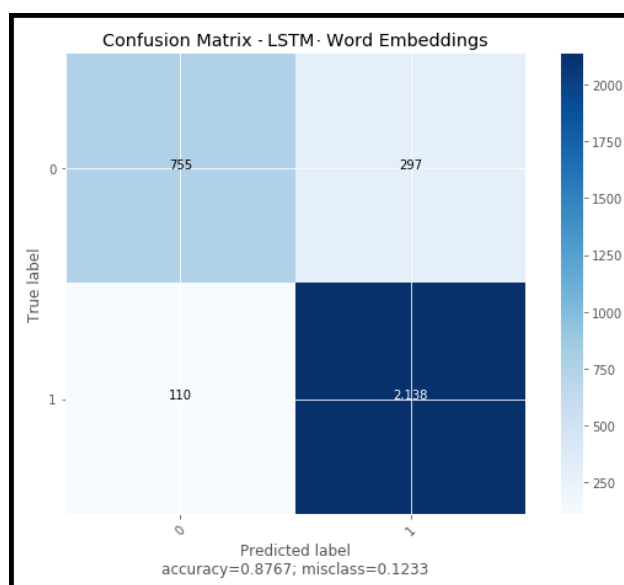
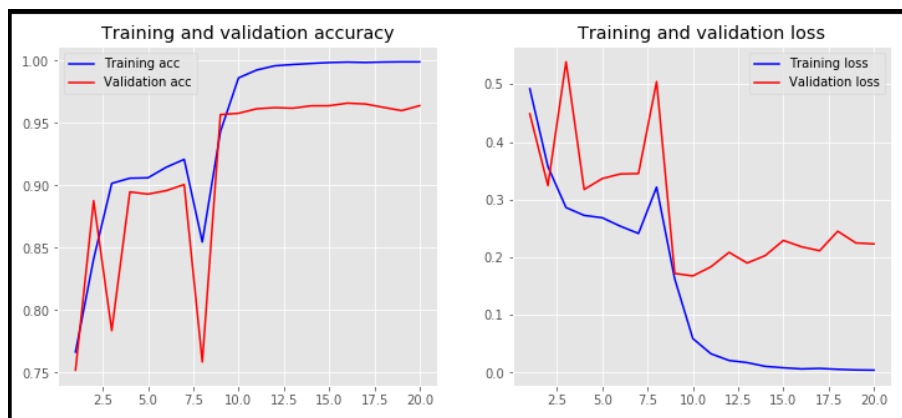


Figura 26 - LSTM com 10.000 dados: matriz de confusão

Fonte: Própria (2018)

Verifica-se que de 1.052 notícias falsas, 755 foram classificadas corretamente e 297 não o foram. Já para as notícias verdadeiras, nota-se que de 2.248, temos que 2.138 estão classificadas corretamente e 110 não estão.

Para o último caso (*dataset* grande), o gráfico a seguir mostra que inicialmente os valores de acurácia não são consistentes, oscilando bastante. Porém, a partir de *epochs* = 9, essa condição some, resultando em valores que variam menos.

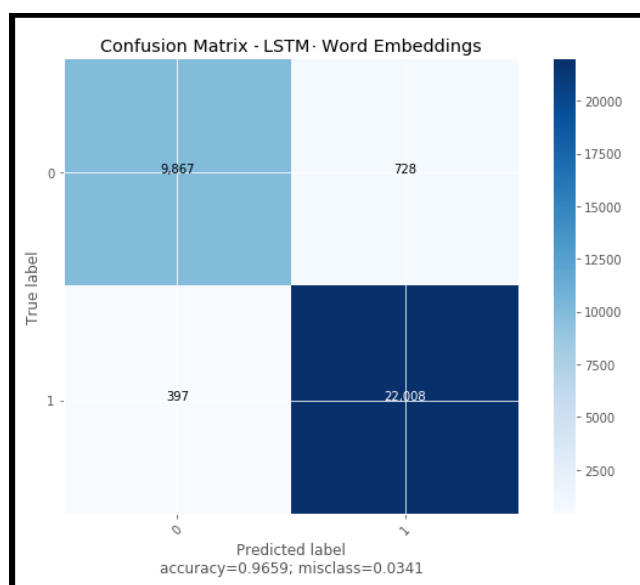


**Gráfico 18 - LSTM com 100.000 dados: gráficos de acurácia e perda**

Fonte: Própria (2018)

A melhor acurácia obtida se dá em  $epochs = 17$ , onde a mesma é igual a 97%, aproximadamente. Isto demonstra que somente para bases com grandes quantidades de dados, o modelo *LSTM* consegue produzir resultados satisfatórios. Já o menor erro no conjunto de validação se deu em  $epochs = 9$ .

Abaixo, a figura mostra a matriz de confusão para o *dataset* grande (cem mil dados). O erro de classificação de 3% mostra que este modelo produz resultados confiáveis.



**Figura 27 - LSTM com 100.000 dados: matriz de confusão**

Fonte: Própria (2018)

De 10.595 notícias falsas, 9.867 tiveram suas classificações dadas de maneira correta e 728 não tiveram. Já para as 22.405 notícias verdadeiras, 22.008 foram classificadas corretamente e 397 não o foram.

#### 4.3. COMPARAÇÃO DOS RESULTADOS

A análise dos modelos de abordagens tradicionais de Aprendizado de Máquina e abordagens de Deep Learning mostrou que algumas destas possuem melhor desempenho do que outras. A tabela a seguir explicita as acurácias obtidas.

	<b>MNB</b>		<b>SVM</b>		<b>CNN</b>	<b>LSTM</b>
	<i>Tf-idf</i>	<i>BoW</i>	<i>Tf-idf</i>	<i>BoW</i>	<i>Word Embeddings</i>	
<b>Dataset Pequeno</b>	89,09%	85,76%	90%	89,09%	91,52%	83,33%
<b>Dataset Médio</b>	89,18%	86,18%	92,48%	91,21%	97,09%	87,67%
<b>Dataset Grande</b>	88,67%	85,69%	93,98%	93%	98,39%	96,59%

Tabela 6 - Relação de acurácias de acordo com o *dataset*

Fonte: Própria (2018)

Pode-se verificar que com exceção do modelo *Multinomial Naive Bayes*, todos os outros obtiveram melhor desempenho utilizando o *dataset* grande (cem mil dados). Logo, *Multinomial Naive Bayes* não é a melhor opção para uma base contendo muitos dados.

Ainda em relação ao modelo *Multinomial Naive Bayes*, pode-se verificar que ele atua melhor sobre bases de dados médias, embora para as outras os resultados não tenham sido muito diferentes. Com acurácias menores do que 90% em todos os casos, pode-se concluir que para o problema de identificação de *fake news*, este modelo não é o ideal. Contudo, verifica-se que *MNB* possui melhor performance utilizando o extrator de características *Tf-idf*, para todos os *datasets*.

Para *Support Vector Machines*, é possível ver que sua atuação é melhor sobre as bases de dados utilizando *Tf-idf*, assim como no caso do *MNB*. O desempenho obtido utilizando tanto *Tf-idf* quanto utilizando *BoW* não são ruins, com exceção do caso em que se utiliza *BoW* para a base de dados pequenas, que chega

a ter acurácia menor que 90%. *LSTM* também é uma opção a se considerar (apenas se tratando de bases de dados grandes).

Pode-se notar então que o extrator de características *Tf-idf* é um diferencial para os modelos da abordagem tradicional de *Aprendizado de Máquina*, gerando melhores resultados. Ainda, concluímos que a melhor opção para identificação de *fake news* utilizando essa abordagem, é com o modelo *SVM*, para todos os tamanhos de *dataset*.

Em relação às abordagens de *Deep Learning*, fica claro que o modelo *CNN* é superior ao *LSTM*, quando se trata da identificação de notícias falsas, com destaque para a utilização do mesmo com o *dataset* grande, que teve sua acurácia igual a 98%, sendo esta a melhor de todos os modelos. Percebe-se também que para as bases de dados pequena e média, o melhor desempenho também foi utilizando a rede neural de convolução.

Dessa maneira, chegamos à conclusão de que para a resolução do problema da identificação de *fake news*, a melhor opção em todos os casos é utilizando *CNN*. Dessa forma, a abordagem de *Deep Learning* é superior às abordagens de método tradicional de *Aprendizado de Máquina*. Porém, utilizar o modelo *SVM* com *Tf-idf*, também se torna uma ótima opção, dados os valores de acurácia obtidos.

## 5. CONCLUSÃO

Seguindo uma abordagem experimental e exploratória, este trabalho foi desenvolvido com o intuito de avaliar o desempenho de métodos tradicionais de Aprendizado de Máquina e de Deep Learning, a fim de resolver o problema da identificação das chamadas *Fake News*. Foram utilizados os modelos *Multinomial Naive Bayes* e *Support Vector Machines* que são referentes à abordagem tradicional de Aprendizado de Máquina, junto com os extratores de características *Term frequency-inverse document frequency* e *Bag-of-Words*. Também foram utilizados os modelos *Convolutional Neural Network* e *Long Term-Short Memory*, referentes à abordagem de Deep Learning.

Em relação a estes modelos, foram realizadas buscas sobre determinados parâmetros que influenciam diretamente no resultado final, que consiste na acurácia. Por fim, foi realizada uma comparação dos resultados, para que fosse possível visualizar qual possui melhor performance dentre a classe de abordagem a qual ele pertence e qual possui melhor performance independente de classe de abordagem.

### 5.1. LIMITAÇÕES

As limitações encontradas se deram por conta do poder de processamento. Treinar os modelos da abordagem tradicional foi uma operação muito custosa e exigiu bastante tempo, principalmente em relação à base de dados contendo cem mil notícias. Não foi possível treinar o modelo *LSTM*, da abordagem de Deep Learning, no computador pessoal do autor deste trabalho, devido à lentidão da execução do mesmo. Dessa forma, foi necessário alugar uma máquina virtual na suíte de computação em nuvem Google Cloud, para dispor de poder de processamento de GPU. Com essa opção em mãos, o treinamento das redes neurais ocorreu de forma rápida. A última limitação se deu por conta da grande quantidade de parâmetros existentes na maioria dos modelos. Realizar testes em todos estes parâmetros, em busca do melhor modelo possível, exigiria executar uma combinação de possibilidades muito grandes, sendo esta uma operação custosa que demandaria um enorme poder computacional. Sendo assim, os parâmetros que



foram modificados nos modelos, foram os que são mais utilizados de acordo com as informações encontradas nos diversos meios de informação.

## 5.2. TRABALHOS FUTUROS

Como trabalho futuro sugere-se a investigação mais profunda de outros parâmetros que afetem diretamente na acurácia dos modelos investigados. Ainda, uma notícia possui mais classificações do que apenas o estado binário estudado no mesmo (real/falsa), como sátira, notícias de ódio, teoria da conspiração, dentre outros. Sugere-se então o estudo de classificação multiclasse para um conjunto de notícias.

## REFERÊNCIAS BIBLIOGRÁFICAS

ALASADI, S.; BHAYA, W. Review of Data Preprocessing Techniques in Data Mining. **ResearchGate**, 2017. Disponível em: <[https://www.researchgate.net/publication/320161439\\_Review\\_of\\_Data\\_Preprocessing\\_Techniques\\_in\\_Data\\_Mining](https://www.researchgate.net/publication/320161439_Review_of_Data_Preprocessing_Techniques_in_Data_Mining)>. Acesso em: 19 dez. 2018.

CAJÚ, L. As fake news e o panoptismo de Michel Foucault. **Ciberjor**, 2017. Disponível em: <<http://www.ciberjor.ufms.br/ciberjor8/files/2017/08/As-fake-news-e-o-panoptismo-de-Michel-Foucault.pdf>>. Acesso em: 19 dez. 2018.

CHOLLET, F. **Deep Learning with Python**. [S.l.]: Manning Publications, 2018.

DFNDR LAB. **Relatório da Segurança Digital no Brasil**. [S.l.]. 2018.

GOLDBERG, Y. **Neural Network Methods for Natural Language Processing**. 1ª. ed. Toronto: Morgan & Claypool Publishers, 2017.

GOMES, H. Facebook e Google miram modelo de negócio das notícias falsas; entenda. **G1**, 2017. Disponível em: <<https://g1.globo.com/e-ou-nao-e/noticia/facebook-e-google-miram-modelo-de-negocio-das-noticias-falsas-entenda.ghtml>>. Acesso em: 19 dez. 2018.

GRIGOREV, A. **Mastering Java for Data Science**. [S.l.]: Packt Publishing, 2017.

IKONOMAKIS, E.; KOTSIANTIS, S.; TAMPAKAS, V. Text Classification Using Machine Learning Techniques. **ResearchGate**, 2005. Disponível em: <[https://www.researchgate.net/publication/228084521\\_Text\\_Classification\\_Using\\_Machine\\_Learning\\_Techniques](https://www.researchgate.net/publication/228084521_Text_Classification_Using_Machine_Learning_Techniques)>. Acesso em: 20 ago. 2018.

ITU. Measuring the Information Society Report 2015. **International Telecommunication Union**, 2015. Disponível em: <<https://www.itu.int/en/ITU-D/Statistics/Documents/publications/misr2015/MISR2015-ES-E.pdf>>. Acesso em: 19 dez. 2018.

KIM, Y. Convolutional Neural Networks for Sentence Classification. **Association for Computational Linguistics**, 2014. Disponível em: <<http://www.aclweb.org/anthology/D14-1181>>. Acesso em: 10 set. 2018.

MANNING, C.; RAGHAVAN, P.; SCHÜTZE, H. **An introduction to Information Retrieval**. Cambridge: Cambridge University Press, 2009.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of Machine Learning**. Massachussets: The MIT Press, 2012.

MURPHY, K. **Machine Learning: A Probabilistic Perspective**. 1ª. ed. Massachussets: The MIT Press, 2012.

NORVIG, P. **Inteligência Artificial**. Tradução de Regina SIMILLE. 3ª. ed. Rio de Janeiro: Elsevier, 2013.

NOWAK, J.; TASPINAR, A.; SCHERER, R. LSTM Recurrent Neural Networks for Short Text and Sentiment Classification. **ResearchGate**, 2017. Disponível em: <[https://www.researchgate.net/publication/318018787\\_LSTM\\_Recurrent\\_Neural\\_Networks\\_for\\_Short\\_Text\\_and\\_Sentiment\\_Classification](https://www.researchgate.net/publication/318018787_LSTM_Recurrent_Neural_Networks_for_Short_Text_and_Sentiment_Classification)>. Acesso em: 14 nov. 2018.

NVIDIA. Recurrent Neural Network. **NVIDIA Developer**, 2018. Disponível em: <<https://developer.nvidia.com/discover/recurrent-neural-network>>. Acesso em: 01 dez. 2018.

PORCELLO, F.; BRITES, F. Verdade x Mentira: A ameaça das fakenews nas eleições de 2018 no Brasil. **Portal Intercom**, 2018. Disponível em: <<http://portalintercom.org.br/anais/nacional2018/resumos/R13-0364-1.pdf>>. Acesso em: 19 dez. 2018.

ROBINSON, D.; SILGE, J. **Text Mining with R**. [S.l.]: O'Reilly Media, 2017.

SARKAR, D. **Text Analytics with Python**. [S.l.]: Apress, 2016.

SECOM. Pesquisa Brasileira de Mídia 2016 - Hábitos de Consumo de Mídia pela População Brasileira. **Pesquisa Brasileira de Mídia**, 2016. Disponível em: <[http://pesquisademidia.gov.br/files/E-Book\\_PBM\\_2016.pdf](http://pesquisademidia.gov.br/files/E-Book_PBM_2016.pdf)>. Acesso em: 02 dez. 2018.

THOTA, A. et al. Fake News Detection: A Deep Learning Approach. **SMU Scholar**, 2018. Disponível em:

<<https://scholar.smu.edu/cgi/viewcontent.cgi?article=1036&context=datasciencereview>>. Acesso em: 19 dez. 2018.

VANDERPLAS, J. **Python Data Science Handbook**. 1<sup>a</sup>. ed. Califórnia: O'Reilly Media, 2017.