**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**

**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.

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Profº. Dr. José Viterbo Filho  
Orientador  
UFF

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Profª. Dra. Aline Marins Paes Carvalho  
UFF

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Profº. Leonardo Pio Vasconcelos  
UFF

NITERÓI - RJ2018 **AGRADECIMENTOS**

Lorem ipsu.

LISTA DE FIGURAS

[Figura 1 - Exemplo de classificação binária 9](#_Toc532394620)

[Figura 2 - Exemplo de tokenização 11](#_Toc532394621)

[Figura 3 - Exemplo de palavras de parada 12](#_Toc532394622)

[Figura 4 - Fórmula Tf-idf 15](#_Toc532394623)

[Figura 5 - Teorema de Bayes 16](#_Toc532394624)

[Figura 6 - Hiperplanos que dividem com sucesso o classificador *SVM* 18](#_Toc532394625)

[Figura 7 - Hiperplano ótimo - *SVM* 19](#_Toc532394626)

[Figura 8 – Contexto no qual Deep Learning está inserido 20](#_Toc532394627)

[Figura 9 - Rede Neural Simples x Rede Neural Profunda 21](#_Toc532394628)

[Figura 10 - Convolução 1D 23](#_Toc532394629)

[Figura 11 - Fórmula Max-pooling 24](#_Toc532394630)

[Figura 12 - Rede Neural de Recorrência 25](#_Toc532394631)

[Figura 13 - Modelo *LSTM* 26](#_Toc532394632)

LISTA DE TABELAS

[Tabela 1 - Matriz Documento-Termo (*Bag of Words*) 14](#_Toc532394647)

[Tabela 2 - Matriz Documento-Termo (*Tf-idf*) 15](#_Toc532394648)

[Tabela 3 – Estrutura da base de dados utilizada 28](#_Toc532394649)

**SUMÁRIO**

[1. introduÇÃO 1](#_Toc532394538)

[1.1. DEFINIÇÃO DO PROBLEMA 3](#_Toc532394539)

[1.2. OBJETIVOS 4](#_Toc532394540)

[1.2.1. OBJETIVOS ESPECÍFICOS 5](#_Toc532394541)

[2. CONCEITOS BÁSICOS 7](#_Toc532394542)

[2.1. APRENDIZADO DE MÁQUINA 7](#_Toc532394543)

[2.2. APRENDIZADO SUPERVISIONADO 7](#_Toc532394544)

[2.2.2. CLASSIFICAÇÃO 8](#_Toc532394545)

[2.3. PROCESSAMENTO DE LINGUAGEM NATURAL 9](#_Toc532394546)

[2.3.1. PRÉ-PROCESSAMENTO DOS DADOS 10](#_Toc532394547)

[2.3.2. EXTRAÇÃO DE CARACTERÍSTICAS DE TEXTOS 13](#_Toc532394548)

[2.4. CLASSIFICAÇÃO *NAIVE BAYES* 16](#_Toc532394549)

[2.5. CLASSIFICAÇÃO *SUPPORT VECTOR MACHINEs* 17](#_Toc532394550)

[2.6. APRENDIZADO PROFUNDO 19](#_Toc532394551)

[2.7. REDE NEURAL DE CONVOLUÇÃO 22](#_Toc532394552)

[2.8. REDE NEURAL DE RECORRÊNCIA 24](#_Toc532394553)

[2.8.1. LONG SHORT-TERM MEMORY 25](#_Toc532394554)

[4. METODOLOGIA 27](#_Toc532394555)

[7. REFERÊNCIAS BIBLIOGRÁFICAS 30](#_Toc532394556)

# 1. introduÇÃO

Ao final do século XX e ao longo do século XXI, o número de usuários que utilizam e possuem internet em suas casas tem crescido bastante, ano após ano. “Em 2000, os internautas eram 6,5% da população mundial. Em 2015, esse índice subiu para 43%. A proporção de casas com conexão à rede chegou a 46% no ano. Essa porcentagem é maior na Europa (82,1%) e menor na África (10,7%).” (G1, 2015). Não é incomum o fato de que a porcentagem de utilizadores 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%). .

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 utilizemos a internet massivamente como fonte de informação, não só pela facilidade, mas também pela sua utilidade.

Desde as eleições presidenciais nos Estados Unidos, em 2016, o fenômeno social conhecido como *Fake News*, que foi utilizado para prejudicar ambas as campanhas,tem sido destaque na imprensa mundial. De acordo com Cunha (2018) em relação às *Fake News:*

A ampla facilidade de acesso, o incremento e a popularização de mecanismos para a criação de conteúdo, aliados à extensão espacial que as mencionadas redes sociais podem conferir ao conteúdo produzido criaram um fenômeno curioso: a elaboração de peças ditas informativas sem critério jornalístico e muitas vezes sem nenhuma correspondência na realidade. Trata-se do que se convencionou chamar fake news (notícias falsas).

De acordo essa explicação, devido ao fácil acesso nos dias de hoje a produtores de conteúdo, como a internet, ficou mais fácil gerar notícias falsas e espalhá-las. Dessa forma, o uso indevido de tecnologias pode prejudicar nossas vidas e afetar fatos históricos, como as eleições nos Estados Unidos.

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.” . 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:

[...] além da finalidade puramente comercial, as fake news podem ser usadas apenas para criar boatos e reforçar um pensamento, por meio de mentiras e da disseminação de ódio. Dessa maneira, prejudicam-se pessoas comuns, celebridades, políticos e empresas. .

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 Batista (2018), as empresas Google e Facebook divulgaram que irão combater tal fenômeno, devido ao seu crescimento espantoso, no intuito de frear a dissipação de informações falsas em um universo que é tão difícil de monitorar como o da internet. 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 *Machine Learning* (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. Em relação à essas tecnologias:

Empresas [...] estão construindo e treinando modelos de Machine Learning que “leem” artigos denunciados como sendo Fake News e com isso aprendem a identificar certos padrões comuns nesse tipo de atividade. [...] vale ressaltar que a PLN, [...] é principal técnica de identificação de notícias falsas. Este é um campo em rápida expansão. Temos como exemplo o Google Tradutor, que nos últimos anos apresentou grandes avanços, justamente fazendo uso das pesquisas mais recentes na área de Processamento de Linguagem Natural. (LUZ, 2018).

Conforme explicado, esse problema tem chamado a atenção de grandes empresas, que estão investindo na resolução desse fenômeno através de subáreas da inteligência artificial, que estão em alta expansão no momento, chamando a atenção de mais estudiosos.

## 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 *Machine Learning*, 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 classificado e a partir daí é possível predizer 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.

## 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 aprendizado profundo (*deep learning*) a fim de resolver o problema de identificação de *fake news*. Serão treinados quatro algoritmos de aprendizado de máquina, sendo eles: *Support Vector Machine*, *Naive Bayes*, *Convolutional Neural Network* e *Recurrent Neural Network*.

### OBJETIVOS ESPECÍFICOS

* Selecionar *dataset* contendo um conjunto de notícias verdadeiras e falsasque possa ser utilizado no problema de classificação de textos.
* Verificar dentre os parâmetros mais utilizados, quais geram melhor acurácia para o classificador *Naive Bayes* com o modelo *bag-of-words* (*BoW*)e com o modelo *term frequency-inverse document frequency* (*tf-idf*).
* Realizar uma comparação com base em acurácia utilizando o classificador *Naive Bayes* com o modelo *bag-of-words* (*BoW*)e com o modelo *term frequency-inverse document frequency* (*tf-idf*).
* Verificar dentre os parâmetros mais utilizados, quais geram melhor acurácia para o classificador *Support Vector Machine* (*SVM*) com o modelo *bag-of-words* (*BoW*)e com o modelo *term frequency-inverse document frequency* (*tf-idf*).
* Realizar uma comparação com base em acurácia utilizando o classificador *Support Vector Machine* (*SVM*) com o modelo *bag-of-words* (*BoW*)e com o modelo *term frequency-inverse document frequency* (*tf-idf*).
* Verificar de acordo com os parâmetros mais utilizados e de acordo com a quantidade de camadas, quais geram melhor acurácia para a classe de rede neural artificial *Recurrent Neural Network* (*RNN*).
* Verificar de acordo com os parâmetros mais utilizados, quais geram melhor acurácia para a classe de rede neural artificial *Recurrent Neural Network* (*RNN*).
* Verificar de acordo com os parâmetros mais utilizados, quais geram melhor acurácia para a classe de rede neural artificial *Convolutional Neural Network* (*CNN*).
* Realizar uma comparação com base em acurácia utilizando as classes de redes neurais artificiais *Recurrent Neural Network* (*RNN*) 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 1.000 dados.
* Verificar qual classificador ou classe de rede neural possui melhor acurácia com um *dataset* contendo 10.000 dados.
* Verificar qual classificador ou classe de rede neural possui melhor acurácia com um *dataset* contendo 100.000 dados.
  1. ORGANIZAÇÃO

**[ESCREVER MAIS TARDE]**

# CONCEITOS BÁSICOS

## 2.1. APRENDIZADO DE MÁQUINA

Podia parecer algo impossível de se imaginar há 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 (*Machine Learning*). “Aprendizado de Máquina consiste em desenvolver algoritmos de previsões eficientes e precisas.” (MOHRI; ROSTAMIZADEH; TALWALKAR, 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 a olho nu talvez fosse difícil de ver, 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; ROSTAMIZADEH; TALWALKAR, 2012). 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.

### 2.2.2. 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 **x** para as saídas , onde **∈ {1,...,C}**, com **C** sendo o número de classes. Se **C = 2**, chamamos de classificação binária (caso em que muitas vezes assumimos **y ∈ {0,1}**); [...] Assumimos  **= (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 **= (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 (círculo e cruz). Também existem duas entradas desconhecidas (representadas por círculos amarelos) em cada um desses conjuntos.

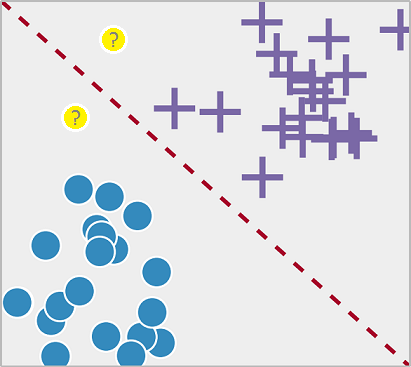


Figura - Exemplo de classificação binária

Fonte:

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.

## 2.3. 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, com o objetivo de que os computadores consigam interpretar, entender, extrair informações e analisar (de maneira sintática, semântica, léxica e morfológica) textos, através do seu processamento.

### 2.3.1. PRÉ-PROCESSAMENTO DOS DADOS

Quando estamos trabalhando com aplicações PLN, existe a necessidade de transformar textos brutos em algo no qual o modelo de *Machine Learning* 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:

Pré-processamento de dados é o processo de limpar e preparar o texto para classificação. O texto usualmente contém algum ruído e dados menos significativos. Isso precisa ser identificado e limpo para criar classificadores mais fortes, rápidos e precisos. Se tais palavras insignificantes não forem removidas, isso aumentaria a complexidade, adicionando mais dimensões na extração de recursos. (KĘSIK, 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 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; RAGHAVAN; SCHÜTZE, 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*.

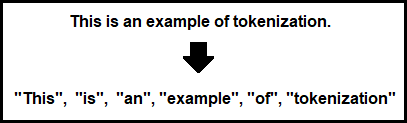


Figura - Exemplo de tokenização

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. “Muitas vezes, isso é uma boa idéia: permitirá que instâncias de Automóveis no início de uma sentença correspondam a uma consulta de automóvel.” (MANNING; RAGHAVAN; SCHÜTZE, 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 remover tais entidades, pois elas não nos dizem nada de relevante ao texto. “Geralmente, as tags HTML são um desses componentes que não agregam muito valor à compreensão e análise de texto.” (SAKAR, 2018). 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; RAGHAVAN; SCHÜTZE, 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; RAGHAVAN; SCHÜTZE, 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.

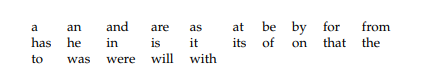


Figura - Exemplo de palavras de parada

Fonte:

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.

### 2.3.2. 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 Análise de Texto é um importante campo de aplicação para algoritmos de aprendizado de máquina. Apesar dos dados brutos, uma seqüência de símbolos não pode ser alimentada diretamente para os próprios algoritmos como a maioria deles espera vetores de características numéricas com um tamanho fixo em vez dos documentos de texto bruto com comprimento variável. (SCIKIT-LEARN, 2018, 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.” . 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.

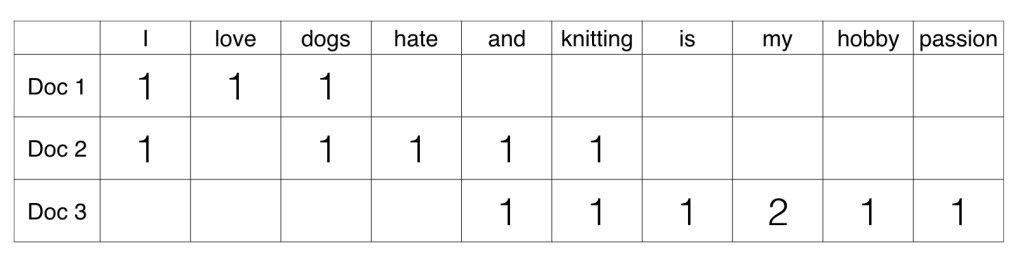


Tabela - Matriz Documento-Termo (*Bag of Words*)

Fonte:

. 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. “*Term frequency* mede a importância local de uma palavra. Se a palavra aparece muitas vezes, então a palavra deve ser importante.” (CALDERON, 2017, tradução nossa). 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 considera uma palavra de assinatura do documento. “[...] a frequência do documento de uma palavra de assinatura deve ser baixa, o que significa que a frequência do documento inverso deve ser alta.” (CALDERON, 2017, tradução nossa). 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 fórmula 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 logarítmo da divisão do número total **N** de documentos pelo número de documentos para qual **x** ocorre.

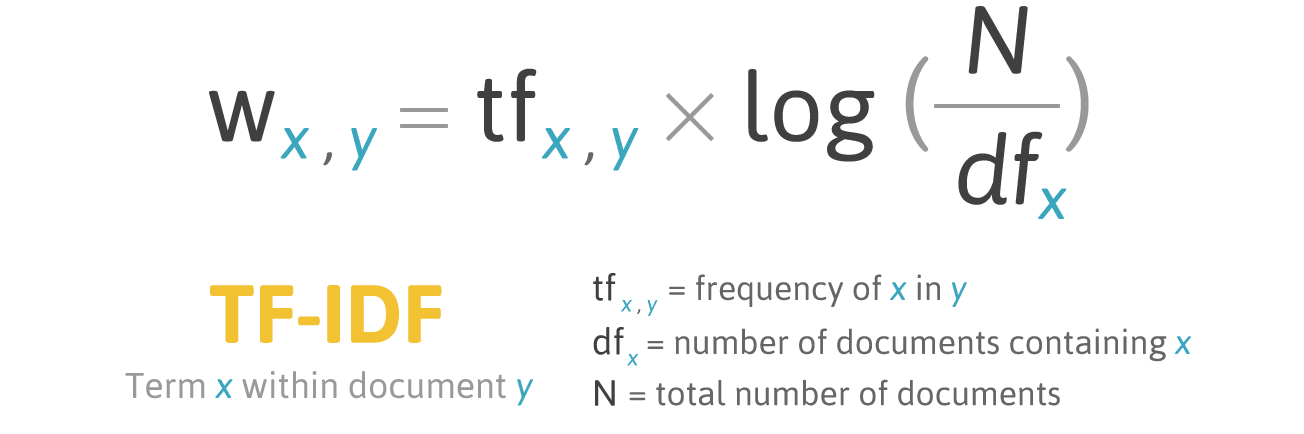


Figura - Fórmula Tf-idf

Fonte:

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.

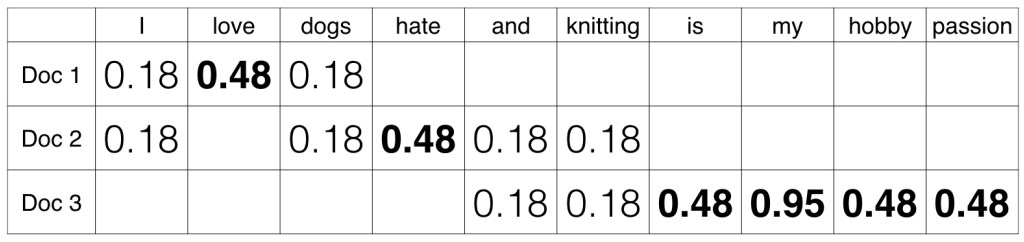


Tabela - Matriz Documento-Termo (*Tf-idf*)

Fonte:

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 afirma que:

[...] *word embeddings* não entende o texto como um humano faria, mas mapeia a estrutura estatística da linguagem usada no corpus. Seu objetivo é mapear o significado semântico em um espaço geométrico. Este espaço geométrico é então chamado de *embedding space* [...]. Se o *embedding* captura bem a relação entre as palavras, coisas como a aritmética vetorial devem se tornar possíveis. Um exemplo famoso neste campo de estudo é a capacidade de mapear *King* - *Man* + *Woman* = *Queen*. (JANAKIEV, 2018, 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.

## 2.4. 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 figura abaixo, temos o teorema de Bayes. Ele consiste em calcular a probabilidade de um evento **y** (features) acontecer dado que **x** aconteceu (L).

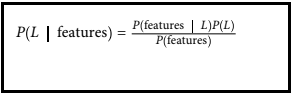


Figura - Teorema de Bayes

Fonte: VANDERPLAS

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.

## 2.5. CLASSIFICAÇÃO *SUPPORT VECTOR MACHINEs*

*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).

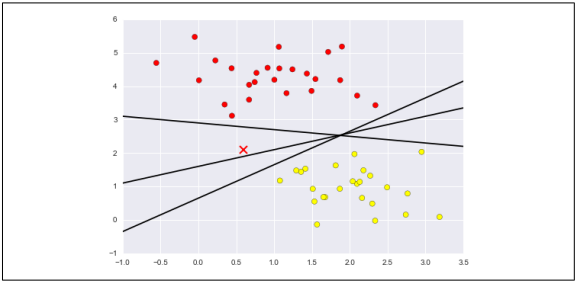


Figura - Hiperplanos que dividem com sucesso o classificador *SVM*

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 (GANDHI, 2018, tradução nossa).

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

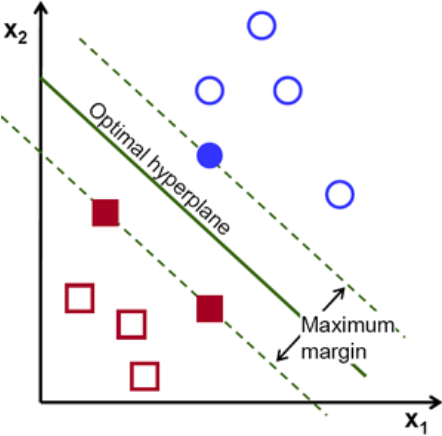


Figura - Hiperplano ótimo - *SVM*

Fonte: (GANDHI, 2018)

Tais pontos são chamados de vetores de suporte. “Os vetores de suporte são pontos de dados mais próximos do hiperplano e influenciam a posição e orientação do hiperplano. Usando esses vetores de suporte, maximizamos a margem do classificador.” (GANDHI, 2018, 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*.

## 2.6. APRENDIZADO PROFUNDO

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

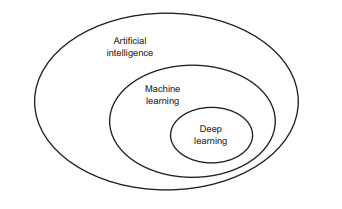


Figura – Contexto no qual Deep Learning está inserido

Fonte:

Com isso, nota-se que Deep Learning é uma área do vasto campo de Aprendizado de Máquina. Apesar de ser uma subárea de *Machine Learning*, a forma de aprender no Aprendizado Profundo 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 aprendizado profundo:

No aprendizado profundo, 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. .

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 aprendizado profundo.

Abaixo, temos a representação de uma rede neural simples e uma rede neural profunda. Percebe-se que a grande diferença entre as duas está no número de camadas ocultas.

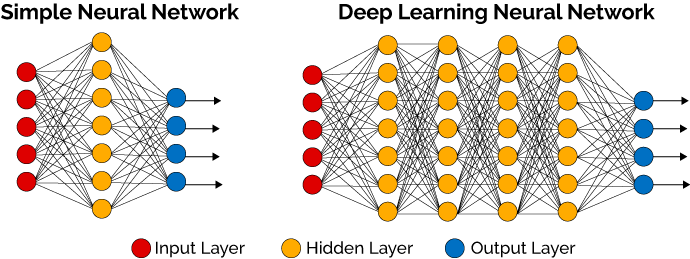


Figura - Rede Neural Simples x Rede Neural Profunda

Fonte:

Uma rede neural simples consiste apenas de uma camada de entrada, oculta e de saída, enquanto que a rede neural profunda consiste de uma camada de entrada, mais de uma camada oculta e uma de saída. Dessa forma, para se caracterizar rede neural profunda, percebe-se que é necessário possui mais de uma camada oculta.

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. .

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 . 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 Aprendizado Profundo em relação aos métodos tradicionais de Aprendizado de Máquina.

## 2.7. 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. Sua importância para essa área se dá ao fato de ter como atributo forte a habilidade de encontrar padrões, sendo assim um importante extrator de características (JANAKIEV, 2018). 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 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 convoluçao, onde temos um filtro (Kernel Size) de tamanho igual a cinco, de uma dimensão, junto com *pooling*, que é utilizado na rede neural de convolução.

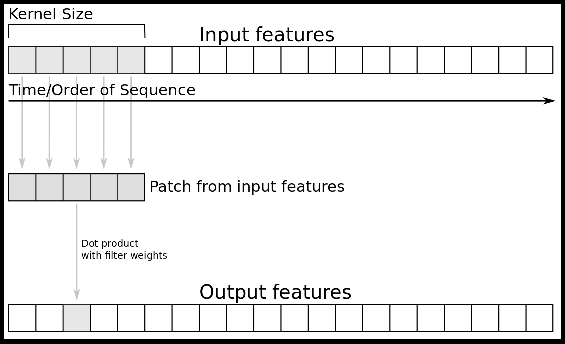


Figura - Convolução 1D

Fonte: (JANAKIEV)

Inicialmente temos um vetor de palavras que juntos formam uma sentença e utilizaremos todas elas em partes, através do tamanho do filtro (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 deslizante 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 figura a seguir traz o conceito de Max-pooling, um dos tipos de *pooling* utilizados após a convolução.

maxpooling.png

Figura - Fórmula Max-pooling

Fonte: (GOLDBERG, 2017)

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).

## 2.8. REDE NEURAL DE RECORRÊNCIA

Com a Rede Neural de Recorrência (*Recurrency Neural Network* ou *RNN*), é possível obter o relacionamento de uma palavra em uma sentença com tudo o que já fora processado antes da ocorrência da mesma (ainda na mesma sentença). “Devido a sua memória interna, as RNNs conseguem lembrar de coisas importantes sobre as entradas recebidas, o que permite que elas sejam muito precisas na previsão do que está por vir.” (DONGES, 2018). Diferentemente da estrutura observada em uma *CNN*, a Rede Neural de Recorrência 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 **Xi** éuma entrada de dados e **hi**a saída, onde **0 ≤ i ≤ t** e **A** representa a saída do nó imediatamente anterior.

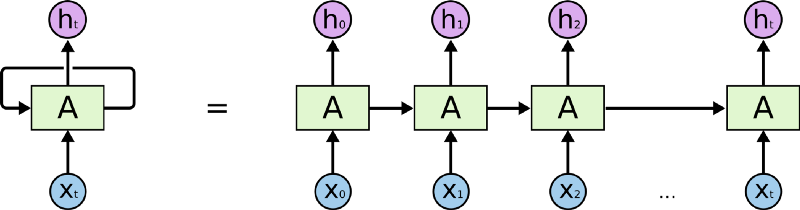


Figura - Rede Neural de Recorrência

Fonte: (DONGES, 2018)

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

### 2.8.1. 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 **si** 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. .

*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.

A figura abaixo mostra o modelo *LSTM*, conforme explicado anteriormente, constituído por portas lógicas nas quais as informações podem passar.

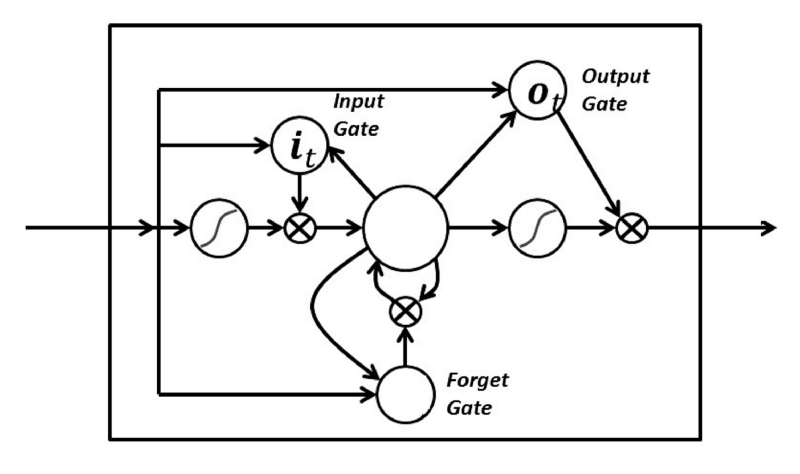


Figura - Modelo *LSTM*

Fonte: (DONGES, 2018)

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 (DONGES, 2018).

# 

# 4. METODOLOGIA

Este trabalho foi desenvolvido utilizando a linguagem Python, através do Jupyter Notebook, que consiste em uma aplicação web *open-source* interativana qual é possível programar de forma organizada e mostrar partes do código de forma eficiente. 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 Aprendizado Profundo 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 NVIDA Tesla P100
* Memória RAM de 30GB
* 10 vCPUs
* HD de 50GB

Foi seguida uma abordagem experimental e exploratória, e desta forma a metodologia deste trabalho consistiu em um estudo que permitiu a implementação dos modelos de classificação tradicionais (*Naive Bayes* e *Support Vector Machine*) e de aprendizado profundo (*RNN* e *CNN*), e depois a realização de um experimento que nos permitisse verificar quais são os melhores parâmetros (dentre os mais usados) 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 estes citados. Todos esses experimentos foram feitos baseados na mesma base de dados, contendo tamanhos diferentes.

Na tabela abaixo se pode verificar a estrutura da base de dados que foi utilizada. Foram produzidas três versões que foram classificadas como: pequena, média e grande.

|  |  |  |  |
| --- | --- | --- | --- |
| Base | Amostras | Amostras de Treinamento | Amostras de Teste |
| Pequena | 1.000 | 670 | 330 |
| Média | 10.000 | 6700 | 3300 |
| Grande | 100.000 | 67000 | 33000 |

Tabela - Dados dos datasets

Fonte: Própria

Cada tabela possui um número de amostras (1.000, 10.000 e 100.000), que foram separadas em amostras de treinamento e de 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.*

Na tabela acima pode-se verificar a estrutura da base de dados que foi utilizada. Foram produzidas três versões: com 100.000, 10.000 e 1.000 dados, que para esse trabalho estão classificadas como seu tamanho sendo grande, médio e pequeno, respectivamente.

Cada *dataset* possui 6 colunas que indicam a identificação, o site, o conteúdo e o título da notícia. Tais bases são versões modificadas do *dataset* criado por Maciej Szpakowski, ex-aluno da Universidade de *Southampton*.[[1]](#footnote-1) A base de dados é *open source*, e até o momento da conclusão deste trabalho possui 9.408.908 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’*.

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, como a *tokenização, lowercasing, lematização,* remoção de *stop words*, pontuação e *tokens* que não consistem de caracteres alfabéticos e de ASCII.

Com os resultados obtidos, foi possível, para cada base de dados, realizar uma análise comparativa dos melhores valores para os parâmetros mais utilizados de acordo com a literatura para cada algoritmo. Para o algoritmo que utiliza o classificador *Naive Bayes*, foram realizados testes sobre os parâmetros ‘alpha’(de 0.01 a 1) e ‘fit\_prior’(*True* ou *False*), e assim foi possível visualizar qual a combinação desses dois 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) também possibilitou verificar qual valor retorna a melhor acurácia. Os resultados obtidos foram colocados em um gráfico de linhas simples. **[FALTA FAZER TESTE NOS MODELOS DE DEEP LEARNING]**

Tanto o classificador *Naive Bayes* quanto o classificador *Support Vector Machine* foram construídos com dois modelos: utilizando *bag-of-words* e *tf-idf*. Vale lembrar, que nesses dois modelos, foi decidido que termosque aparecem em menos de 20% e mais de 60% do documento, são descartados. **[FALTA FALAR DE WORD EMBEDDING]**.

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 como ‘*FAKE*’e o número de noticiais que foram classificadas corretamente e erroneamente como ‘*REAL*’. Assim, foi possível verificar qual classificador 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 modelo, possui a melhor acurácia, de acordo com o tamanho da base de dados. **[FALAR DE RNN E CNN]**

# 7. REFERÊNCIAS BIBLIOGRÁFICAS

BAIA, C. Introdução ao Machine Learning. **Carlos Baia**, 17 jul. 2016. Disponivel em: <http://carlosbaia.com/2016/07/17/introducao-ao-machine-learning/>. Acesso em: 10 set. 2018.

BATISTA, R. Fake News. **Mundo Educação**, 2018. Disponivel em: <https://mundoeducacao.bol.uol.com.br/curiosidades/fake-news.htm>. Acesso em: 10 nov. 2018.

BISHOP, D. Text Analytics – Document Term Matrix. **Darrin Bishop**, 2017. Disponivel em: <https://www.darrinbishop.com/blog/2017/10/text-analytics-document-term-matrix/>. Acesso em: 20 nov. 2018.

CALDERON, P. Bag of Words and Tf-idf Explained. **Data meets Media**, 2017. Disponivel em: <http://datameetsmedia.com/bag-of-words-tf-idf-explained/>. Acesso em: 20 nov. 2018.

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

CUNHA, R. A divulgação de notícias falsas e as possibilidades de responsabilização. **Estadão**, 2018. Disponivel em: <https://politica.estadao.com.br/blogs/fausto-macedo/a-divulgacao-de-noticias-falsas-e-as-possibilidades-de-responsabilizacao/>. Acesso em: 24 ago. 2018.

DATA SCIENCE ACADEMY. Deep Learning Book. **Data Science Academy**, 2018. Disponivel em: <http://deeplearningbook.com.br/o-que-sao-redes-neurais-artificiais-profundas/>. Acesso em: 05 dez. 2018.

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

DONGES, N. Recurrent Neural Networks and LSTM. **Towards Data Science**, 2018. Disponivel em: <https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5>. Acesso em: 03 dez. 2018.

G1. Mundo tem 3,2 bilhões de pessoas conectadas à internet, diz UIT. **G1**, 2015. Disponivel em: <http://g1.globo.com/tecnologia/noticia/2015/05/mundo-tem-32-bilhoes-de-pessoas-conectadas-internet-diz-uit.html>. Acesso em: 23 ago. 2017.

GANDHI, R. Support Vector Machine — Introduction to Machine Learning Algorithms. **Towards Data Science**, 2018. Disponivel em: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>. Acesso em: 30 nov. 2018.

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

JANAKIEV, N. Practical Text Classification With Python and Keras. **Real Python**, 2018. Disponivel em: <https://realpython.com/python-keras-text-classification/#what-is-a-word-embedding>. Acesso em: 22 out. 2018.

KALYANARANGAN, V. Text Clustering : Get quick insights from Unstructured Data 1. **Machine Learning Blogs**, 2017. Disponivel em: <https://machinelearningblogs.com/2017/01/26/text-clustering-get-quick-insights-from-unstructured-data/>. Acesso em: 22 nov. 2018.

KĘSIK, A. Text Classification (Topic Categorization, Spam filtering, etc). **Commonlounge**, 2017. Disponivel em: <https://www.commonlounge.com/discussion/ea97b828a7a24a5ca11b50da6e3b4b09/history>. Acesso em: 20 set. 2018.

LUZ, G. Eleições na Era da Informação: Big Data, Fake News e Barack Obama. **Mundo ia**, 2018. Disponivel em: <http://mundoia.com.br/opiniao/eleicoes-na-era-da-informacao-big-data-fake-news-e/>. Acesso em: 26 ago. 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.

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

SAKAR, D. A Practitioner's Guide to Natural Language Processing (Part I) — Processing & Understanding Text. **Towards Data Science**, 2018. Disponivel em: <https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72>. Acesso em: 02 nov. 2018.

SCIKIT-LEARN. Feature extraction. **scikit-learn**, 2018. Disponivel em: <https://scikit-learn.org/stable/modules/feature\_extraction.html#text-feature-extraction>. Acesso em: 25 nov. 2018.

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

VANDERPLAS, J. **Python Data Science Handbook**. 1ª. ed. Califórnia: O'Reilly, 2017.

1. https://github.com/several27/FakeNewsCorpus [↑](#footnote-ref-1)