Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e de Tecnologia– CCET
Departamento de Computação– DC
Programa de Pós-Graduação em Ciência da Computação– PPGCC

Lord Flaubert Steve Ataucuri Cruz

Enriquecendo a Previsão de Séries Temporais usando Informação Textual

São Carlos - SP, Brasil
Fevereiro/2021

Lord Flaubert Steve Ataucuri Cruz

Enriquecendo a Previsão de Séries Temporais usando Informação Textual

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Diego Furtado Silva

São Carlos - SP, Brasil
Fevereiro/2021

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Dissertação de Mestrado do candidato Lord Flaubert Steve Ataucuri Cruz, realizada em 25/02/2021.

Comissão Julgadora:

Prof. Dr. Diego Furtado Silva (UFSCar)

Prof. Dr. Ricardo Augusto Souza Fernandes (UFSCar)

Prof. Dr. Ricardo Marcondes Marcacini (USP)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

A minhas filhas Camila e Sarah, meu motivo de lutas e sacrifícios.

Agradecimentos

Primeiramente agradeço a Deus por me dar vida e saúde para conseguir minhas metas, foram muitas vezes que quis desistir e mesmo sem emprego, sem uma renda e uma família que alimentar me deu as forças para continuar, acreditando nele consegui emprego e consegui terminar esta meta.

A minha mãe Alexandrina que foi a pessoa que me encorajou sempre a terminar esta meta que um dia eu tracei, ela foi a pessoa mais lutadora que vi na vida, motivo de orgulho e de exemplo. Também queria agradecer a Valeria minha companheira de lutas que me apoio a conseguir esta meta desde que cheguei no Brasil, pelas esperas, os sacrifícios, por as privações e ajuda apesar que não entende de computação e códigos.

Agradeço a minha família Katy, Raquel, Esther, George e pai Flaubert que não estando perto de mim, se preocupavam comigo e me encorajando a terminar esta pesquisa.

Estou grato por minhas enteadas Ana e Sofia, mesmo não entendendo das coisas que eu faço no computador, todos os dias me trouxeram um café, me ajudaram no meu português e me apoiaram a cuidar da Camila, minha filha, quando estava pesquisando e precisava ajuda com brincar com ela, trocar de fralda e às vezes deram banho quando não estava em casa.

O meu orientador Prof. Dr. Diego Furtado Silva pela excelente orientação durante todo o período do meu Mestrado, por ter acreditado em mim, que mesmo em momentos difíceis me ajudou a seguir a frente e não desistiu de me orientar. Apesar de eu ter fugido de aplicar aprendizado de máquina na área da música, me apaixonei mais pelos textos.

Agradecer as pessoas que tivessem gostado muito de me ver alcançar minhas metas e festejar mas já não estão comigo, mas foram partes importantes da minha vida e do meu passado.

Agradeço à agencia Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e também à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelos auxílios financeiros indispensáveis à realização deste trabalho (Processo n. 49096/2018-6).

Resumo

A capacidade de extrair conhecimento e prever tendências de ações é crucial para mitigar os riscos e incertezas dos investidores no mercado. A tendência das ações é afetada pela não linearidade, complexidade, ruído e especialmente, eventos do entorno. Fatores externos, como notícias diárias, tornaram-se um dos principais recursos dos investidores para a tomada de decisões sobre a compra ou venda de ativos. Porém, essas notícias acontecem muito rápido, são milhares de notícias geradas por diferentes sítios web, demorando muito para serem analisadas, o que pode custar milhões de dólares em perdas para seus investidores devido a uma decisão tardia. Abordagens recentes baseadas em modelos de linguagem contextuais transformaram a área de processamento de linguagem natural. No entanto, os modelos de classificação que usam notícias que influenciam as ações lidam com textos não rotulados, desbalanceados e dissimilares. Estudos recentes mostram que a previsão de séries temporais melhora substancialmente ao considerar informações externas. Este trabalho propõe uma metodologia híbrida em três fases, uma para a mineração de notícias, um modelo de representação de características compactas e uma para a previsão de séries temporais, que se fundem para uma previsão mais precisa dos preços. Inicialmente é construído um corpus pequeno a partir da serie temporal. Após isso, utiliza-se uma rotulação baseada em aprendizado semissupervisionado para atribuir rótulos às demais notícias. Na segunda fase, é realizado o processo de mineração de textos com um classificador de novas notícias, cuja saída é alinhada as características da serie temporal, para que o modelo de representação compactada extraia novas características num espaço latente. Finalmente, realizamos a predição dos preços futuros com este conhecimento fundido. Em um estudo de caso com a cripto-moeda Bitcoin, a metodologia proposta alcançou uma diminuição de 1.62% *no porcentagem de erro médio absoluto.*

Palavras-chaves: *análise de sentimento para séries temporais, enriquecendo as séries temporais, computação financeiras, previsão de séries temporais, previsão com aprendizado profundo, previsão de séries temporais com lstm.*

Abstract

The ability to extract knowledge and forecast stock trends is crucial to mitigate investors' risks and uncertainties in the market. The stock trend is affected by non-linearity, complexity, noise, and especially the surrounding events. External factors such as daily news became one of the investors' primary resources for making decisions about buying or selling assets. However, this kind of information appears very fast. There are thousands of news generated by numerous web sources, taking a long time to analyze them, which can cost millions of dollars losses for investors due to a late decision. Recent contextual language models have transformed the area of natural language processing. However, classification models that use news that influence stock values need to deal with the unlabeled, class imbalance, and dissimilar texts. Recent studies show that the prediction of time series substantially improves by considering external information. This work proposes a hybrid methodology with three phases, one for news mining, a model for representation compact features, and the forecast model of time series, which merge for a more accurate prediction of prices. Initially, a small corpus is built using as support the time series. After that, we label the corpus based on semi-supervised learning to assign labels to other unlabeled news. In the second phase, the mining model with a classifier is used, whose output is concatenated with time series features, so the compact model representation extracts new features in a latent space. Finally, we predicted future prices with this fused knowledge. In a case study with Bitcoin cryptocurrency, the proposed methodology achieved a 1.62% *decrease in the mean absolute percentage error.*

Keywords: *sentiment analysis for time series, enrich time series, computational finance, time series forecasting, deep learning forecasting, lstm time series forecasting*

Lista de ilustrações

Lista de tabelas

Lista de siglas

Lista de abreviaturas e siglas

AM	Aprendizado de Máquina
AS	Análise de Sentimentos
AE	AutoEncoders
BoW	Bag of Words
BERT	Bidirectional Encoder Re-presentations from Transformers
DL	Deep Learning
LSTM	Long Short Term Memory
LP	Label Propagation
PLN	Processamento de Língua Natural
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TF-IDF	Term Frequency – Inverse Document Frequency

Sumário

Capítulo 1

Introdução

Apresenta-se neste Capítulo o contexto em que esta dissertação está inserida, evidenciando as razões que impulsionaram a pesquisa para a temática abordada e a motivação e a problemática que deu origem a este projeto de pesquisa. Em seguida, são apresentados os objetivos propostos e a hipótese de pesquisa, além da síntese da metodologia e, por fim, é descrita a organização desta dissertação de mestrado.

1.1 Contexto

Os mercados financeiros são importantes para a economia e organização da sociedade moderna. As atividades financeiras desempenham um papel importante na economia mundial, já que influenciam o desenvolvimento econômico de vários países, atraindo a atenção de pesquisadores e profissionais de negócios por suas habilidades teóricas e aplicações práticas (CAVALCANTE et al., 2016). Exemplos práticos da importância do estudo deste tipo de mercado podem ser encontrados na literatura. Por exemplo, para Vrigazova, Pavlova e Bogdanova (2016), o mercado financeiro dos EUA é líder na economia global, onde acontecimentos como a crise financeira de 2008 podem se espalhar por vários meses em todo o mundo e atingir outros mercados. Um mercado financeiro é qualquer mercado em que compradores e vendedores participam negociando diferentes ativos, como ações, títulos, moedas e derivativos que expressam valor e podem ser convertidos em dinheiro. O sucesso de um investidor neste tipo de mercados globais ou nacionais depende da qualidade da informação que ele precisa para tomar decisões. Devido à sua importância prática, a análise dos movimentos do mercado financeiro tem sido amplamente estudada nas áreas de finanças, engenharia e matemática, devido ao seu potencial ganho financeiro (YOO; KIM; JAN, 2005).

A análise do mercado compreende vários estudos dos atributos e características do mercado que influenciam o preço de um ativo financeiro. O principal objetivo da análise de mercado é compreender os comportamentos do mercado para ajudar ao processo de tomada de decisões. A literatura financeira sugere dois enfoques utilizados para a análise do movimento dos preços: a Análise Fundamentalista e a Análise Técnica. As duas abordagens têm como objetivo principal entender os movimentos do preço e prever as futuras direções do preço. Porém, estas técnicas são diferentes em natureza (CAVALCANTE et al., 2016).

A análise fundamental pode ser definida como aquele estudo que leva em consideração fatores internos e externos da empresa que influenciam o preço, como a situação econômica de uma empresa, fatores econômicos e políticos, estratégias de marketing, as perspectivas para o seu futuro. Os fundamentalistas tentam definir um valor intrínseco de um ativo em base, a valores como: relatórios, balanços e resultados internos de uma empresa.

A análise técnica é baseada no princípio da *The Down Theory* (MURPHY, 1999), onde a previsão do preço está baseada no comportamento de valores passados e, portanto, utiliza o histórico do preço para procurar padrões e indicadores dos futuros movimentos do mercado (TEIXEIRA; OLIVEIRA, 2010). Esta abordagem não leva em consideração características internas e externas de uma empresa no estudo do movimento do preço no mercado. Técnicos acreditam que o preço já inclui toda essa informação fundamentalista, e que a história tende a se repetir. Esse modelo evita a análise de fatores econômicos subjetivos que acontece ao redor da economia, política e notícias (CAVALCANTE et al., 2016). Eles utilizam uma série de ferramentas como gráficos, indicadores técnicos e modelos que monitoram a tendência do preço e volume sobre o tempo.

O mercado de ações pode ser visto como um sistema complexo que recebe muitas informações, desde informações fundamentais como eventos sociopolíticos, notícias do comportamento dos investidores, parcerias entre empresas ou decisões de organismos de controle que afetam os ativos (JOTHIMANI; SHANKAR; YADAV, 2016). Hoje em dia muitos especialistas deste domínio preferem utilizar as duas abordagens. De todas as possibilidades da análise fundamentalista, são utilizados os fatores externos, especialmente as notícias. As notícias se converteram em um dos principais recursos dos investidores para a tomada de decisões de compra ou venda. Porém, estas notícias acontecem muito rápido, existem milhares de notícias geradas por diferentes recursos, tomando muito tempo em fazer uma análise para cada uma, o que pode custar perdas muito prejudiciais para seus investidores pelas decisões tardias. Assim, temos dois tipos de problemas: a grande quantidade de notícias e o tempo para analisar o impacto destas notícias no mercado.

Nos últimos anos, várias abordagens têm sido propostas para resolver o problema da previsão de séries temporais financeiras. Na literatura encontra-se três principais abordagens para a previsão de séries temporais: modelos estatísticos, modelos baseados em aprendizado de máquina e modelos baseados no processo de mineração de texto (CA-

VALCANTE et al., 2016), que utilizam o sentimento presente nos textos financeiros para ajudar a previsão do preço.

Durante a primeira metade deste século, as aplicações estatísticas baseadas em modelos *Autoregressive* (AR), *Autoregressive Integrated Moving Average* (ARIMA) e *Moving Average* (MA) foram consideradas o estado da arte para modelar as séries temporais e previsão, esta categoria assume que os dados seguem uma distribuição normal, são estacionários e lineares (JOTHIMANI; SHANKAR; YADAV, 2016). Esses algoritmos são conhecidos como paramétricos, devido ao fato de necessitarem um domínio sofisticado de parâmetros matemáticos para o modelo (PARMEZAN; SOUZA; BATISTA, 2019). Geralmente, assumem que as séries temporais são geradas a partir de um modelo linear, que assume relações lineares entre os valores passados da variável de previsão e portanto, os padrões não lineares não podem ser capturados por esses modelos (KUMAR; MURUGAN, 2013).

As últimas décadas, teve um incremento nos métodos baseados em Aprendizado de Máquina, especialmente nos métodos de regressão e métodos baseados em dependência temporal (PARMEZAN; SOUZA; BATISTA, 2019). Devido às características das séries temporais e sua não-linearidade, os modelos anteriores tendem a falhar ao serem aplicados nesse domínio. Ao contrário dos métodos estatísticos, os modelos de Aprendizado de Máquina são capazes de capturar as relações não-lineares e não estacionárias, descrevendo as características dos dados sem conhecimento *a priori* da distribuição dos dados (CAVALCANTE et al., 2016).

Entre as técnicas mais utilizadas estão as Redes Neurais Artificiais (*Artificial Neural Networks*, ANN), que têm sido amplamente utilizadas pois são métodos auto adaptativos, orientados a dados e capazes de capturar comportamentos não lineares, sem qualquer suposição estatística sobre os dados e inclusive com modelos híbridos (CAVALCANTE et al., 2016). Também foram utilizadas outras ANNs, como a *Multilayer Perceptron* (MLPs) com algoritmo de *back-propagation*, para aprender as relações entre indicadores técnicos com o objetivo de prever o preço de abertura e fechamento diário com dados da BOVESPA (Martinez et al., 2009). Em Jasemi, Kimiagari e Memariani (2011), é utilizada uma *Feed-forward Neural Network* (FFNN) para desenvolver uma estratégia para aprender padrões de velas japonesas. Embora as ANNs foram amplamente utilizadas para aprender padrões, elas contém limitações devido a natureza das séries temporais financeiras. Essas desvantagens são: atuam como uma caixa preta, tendência de *overfitting*, taxa de convergência lenta e comumente se estagnam em mínimos locais (CHEN; HAO, 2017). Também têm sido aplicados Máquina de Suporte de Vetores (SVM) e técnicas dos vizinhos mais próximos (kNN-TSPI) (PARMEZAN; BATISTA, 2015) e recentemente Mecanismos de Atenção aplicado a séries temporais (QIN et al., 2017), (LAI et al., 2018), (ZHANG et al., 2019).

Um fator comum dessas abordagens é a construção do modelo a partir dos dados

históricos da série temporal, essas técnicas extraem padrões passados e acreditam que voltariam a se repetir no futuro. Há pesquisadores, que motivados pela Análise Fundamentalista, tentam prever os próximos movimentos do mercado por meio da mineração de textos. Para isso, utilizam especificamente as notícias financeiras do entorno, como relatórios, parcerias ou até mesmo as redes sociais são utilizadas para extrair informações relevantes em dados textuais, com o intuito de identificar sentimentos nas notícias para melhorar a previsão do preço.

Este tipo de abordagens que extraem informação qualitativa de notícias, é conhecida como análise de sentimentos (AS). A AS têm sido uma das áreas de pesquisa mais ativas em Processamento de Língua Natural (PLN). Os algoritmos de PLN se beneficiaram muito do amadurecimento das redes neuronais recorrentes. Este tipo de tarefas utilizam modelos de aprendizado profundo baseadas em *embedding* contextuais como o Bert (DEVLIN et al., 2019) e outros, os últimos trabalhos da literatura provaram que esses modelos são úteis para algumas tarefas do PLN, que inclusive ultrapassam o ser humano (SUN et al., 2020; BIESIALSKA; BIESIALSKA; RYBINSKI, 2020; Sousa et al., 2019; LI et al., 2017).

Estudos recentes fornecem evidências que para alguns domínios e cenários, pode ser útil a incorporação de informação adicional do entorno do tipo texto na previsão da série temporal, já que ajudam a melhorar a previsão do preço. Por exemplo, os autores de Rodrigues et al. (2018), incorporam as notícias de agronegócios na previsão do preço do milho, conseguindo melhora da previsão em alguns meses. Em Marcacini, Carnevali e Domingos (2016), mostram que incorporar notícias externas da produção de celulose num algoritmo simples como o KNN pode melhorar a previsão das séries temporais. O trabalho de Koppel e Shtrimberg (2004), analisa como o mercado reage a notícias positivas e negativas para o ativo S&P500.

Ainda são poucos os estudos que usam esse conhecimento externo extraído por meio de um processo de mineração de texto e usam essas informações para enriquecer as séries temporais sem deixar de analisar os padrões passados presentes na análise técnica. Neste trabalho, apresentamos uma metodologia híbrida que funde as duas abordagens da análise de mercados para previsão do preço para um caso de uso do *Bitcoin*. Na primeira fase, realizamos o processo de mineração de textos até a classificação do sentimento, utilizando um corpus parcialmente rotulado com uma estrategia semi automatizada, usamos aprendizado transdutivo para espalhar o rótulo a partir de um conjunto pequeno de textos rotulados a outros textos sem rótulo. Na segunda fase, para enriquecer as séries temporais alinhamos esse conhecimento externo a um conjunto de características abstratas extraídas dos indicadores técnicos, finalmente realizamos a previsão do preço com estas informações.

1.2 Problemática e Motivação

Dentro do contexto apresentado é possível identificar que existem abordagens baseadas em aprendizado de máquina, modelos estatísticos e modelos baseados em dados textuais para a previsão de séries temporais. As duas primeiras abordagens ignoram os eventos externos que ocorrem ao redor de um ativo. A percepção ou opiniões são influenciadores importantes de nosso comportamento humano, eventos externos como: notícias, parcerias firmadas, eventos sociopolíticos, decisões de organizações de controle e outros, influenciam muito nos preços de um ativo do mercado. As notícias se converteram em um dos principais recursos dos investidores para a tomada de decisões de compra ou venda. Porém, estas notícias acontecem muito rápido, existem milhares de notícias geradas por diferentes sítios web, tomando muito tempo para realizar uma análise para cada uma delas, o que pode custar perdas muito significativas para seus investidores pela decisões tardias.

Paralelamente, o investidor precisa analisar padrões passados e gráficos no intuito de encontrar algum tipo de informação útil para a tomada de decisões. Assim, utilizar este tipo de informação textual e dados quantitativos paralelamente convertem numa tarefa de difícil gerenciamento.

Até o presente momento, há poucos trabalhos que incorporam a informações de notícias dentro de um modelo de previsão sem deixar de considerar a informação quantitativa passada para o caso do *Bitcoin*. Neste contexto, surgiu a motivação em aprofundar a pesquisa relacionada sobre modelos de análise de sentimentos e aprendizado profundo para enriquecer com informações externas o modelo de previsão do preço. Dessa forma, fundimos a análise fundamental e a análise técnica numa ferramenta para o usuário.

1.3 Objetivo

O objetivo central deste trabalho é construir um modelo híbrido que permita adicionar o conhecimento externo em forma de notícias na análise do histórico da série temporal, sem necessidade de uma grande quantidade de textos anotados, com a finalidade de melhorar a precisão da previsão do preço.
Os objetivos específicos procurados são:

a) Criar uma ferramenta que permita fundir duas abordagens: a Análise Fundamental e a Análise Técnica.

b) Construir um corpus maior rotulado semi supervisionadamente, a partir de um conjunto pequeno de textos rotulados com a ajuda de um supervisor.

c) Construir uma *word embedding* relacionada ao domínio do ativo *Bitcoin*.

d) Desenvolver uma ferramenta que permita mitigar o risco e evite grandes perdas ao pequeno investidor que realiza operações intra-day.

e) Contribuir, no domínio da análise de séries temporais financeiras, com um modelo de mineração de texto capaz de detectar notícias de interesse não só do ativo digital *Bitcoin*, mas também de outros ativos do cripto-mercado.

1.4 Hipótese de Pesquisa

A principal hipótese perseguida nesta dissertação é a seguinte:

Acredita-se que um modelo que considera as notícias externas e padrões passados dos indicadores técnicos no cálculo da previsão é capaz de melhorar a acurácia da previsão do preço para o *Bitcoin*.

Para verificar a validade dessa hipótese, comparamos o mesmo modelo de previsão de preços em três versões: baseado somente no histórico do preço, baseado em características compactas e que agrega o sentimento do mercado. Caso o terceiro modelo não apresente melhoria na previsão de valores, esta hipótese não será concretizada.

1.5 Organização do Trabalho

O restantes desta dissertação está organizada da seguinte maneira: No Capítulo 2 são apresentados algumas referências aos mercados financeiros e conceitos relacionados a abordagens técnico e fundamental.

No Capítulo 3 são introduzidos conceitos e características relevantes das séries temporais financeiras. No Capítulo 4 e 5 são apresentados os conceitos de Aprendizado de Máquina e conceitos relacionados ao Processamento de Linguagem Natural.

No Capítulo 6 são apresentados os principais trabalhos relacionados na área. No Capítulo 7 é exposta a proposta de trabalho. Finalmente no Capítulo 8 são apresentados os resultados e as análises dos experimentos realizados desta dissertação.

Capítulo 2

O Mercado

Os mercados, de uma forma ou outra, existem há séculos. Existem registros de notas e cheques entre comerciantes e banqueiros que habitavam a Babilônia desde 2000 A.C. O câmbio de mercadorias e participações em viagens mercantis foram negociadas em Ostia, o porto de Roma, no século II DC (BRAUDEL; REYNOLD, 1992). Os mercados estão presentes em nossa sociedade em diferentes épocas e cidades. Eles são importantes porque movimentam nossa economia e o câmbio de mercadorias de diferentes tipos. Neste capítulo, se abordará os mercados financeiros e aspectos relacionados a eles.

2.1 Os Mercados Financeiros

Um mercado financeiro é qualquer lugar onde "compradores" e "vendedores" se reúnem e participam comprando e vendendo diferentes *assets* (do termo em Inglês a este conjunto) ou ativos como: ações, derivados, títulos e moedas (KIRKPATRICK; DAHLQUIST, 2010). O mercado tem como objetivo comercializar e estabelecer o preço de um conjunto de *assets* para seu comércio local ou global.

Nestes mercados atuam vários tipos de perfis: como o investidor a longo prazo e o *day trader*. Os *traders* são pessoas que comercializam os *assets* e têm um amplo domínio da matéria econômica, formação do preço e conhecimento da análise técnica e fundamental. Normalmente, gerenciam as carteiras de grandes instituições como bancos e empresas.

2.1.1 Formação do Preço

Uma negociação nesses mercados é formada por duas ordens: uma para entrar no mercado – "buy" ou "long" (ordem de compra) – e uma para sair do mercado – "sell" ou

"short" (ordem de venda).

As ordens podem ser de diferentes tipos de acordo com o momento da execução e podem ser executadas no melhor preço disponível ou em um preço específico, segundo (ALDRIDGE, 2009) existem:

1. Limit order, ocorre quando se estabelece um preço específico de compra ou venda de um *asset*.

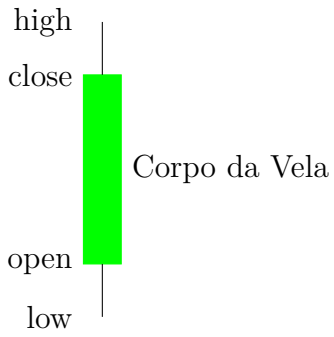
2. Market order, ocorre quando se compra ou vende ao melhor preço atual do mercado.

3. Stop loss, take profit order, é um tipo de ordem que evita as perdas ou ganhos das ordens *limit* ou *market*. Ambas, a *stoploss* e a *take profit* se convertem em ordens *limit* ou *market* se o preço específico de compra ou venda atinge ou passa a um valor conhecido como *stop price*. Este conceito está associado a um risco que o investidor decide passar ao entrar no mercado.

Para determinar o preço de execução, as ordens de compradores e vendedores são colocados em um "livro de ordens" (do termo em inglês *order book*), que ajudam a determinar qual ordem pode ser atendida.

Para descrever os preços, são utilizadas comumente as velas japonesas (*candlesticks*, no termo em inglês) durante um período de tempo. As velas japonesas são compostas pelos preços *Open, High, Close,* e *Low* (OHCL) de um período específico de tempo (15 minutos, 1 hora, 1 dia, 1 semana, etc.). Na Figura 1, mostra-se uma representação do preço em formato de velas japonesas e suas componentes. Essas velas são muito utilizadas para a representação gráfica dos preços e junto a indicadores técnicos da análise técnica (que será introduzida nas subseções seguintes) são ferramentas que os *traders* freqüentemente utilizam.

Figura 1 – Vela Japonesa, formado por Open, High, Close e Low (OHCL).

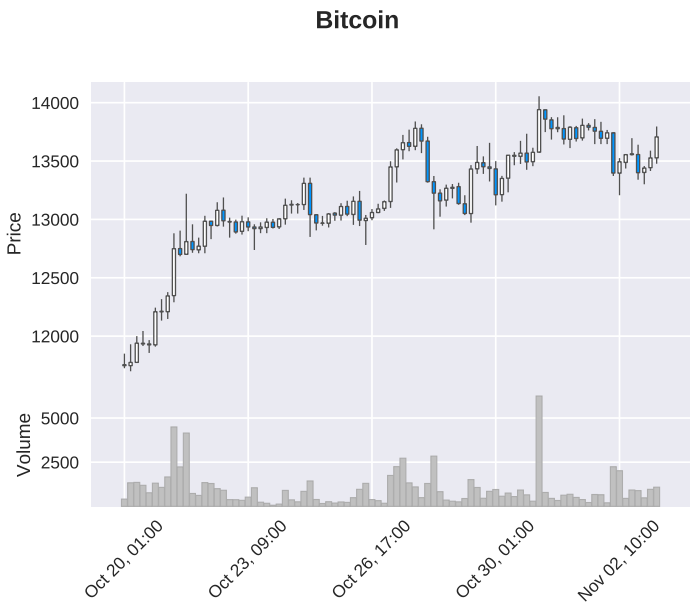


Fonte: Elaborada pelo autor.

Os preços financeiros são formados por várias sequências desse tipo de velas mostradas por um gráfico, as quais são a base para fazer cálculos matemáticos que os partidários da abordagem da Análise Técnica acreditam e inclusive também da Análise Fundamentalista.

Na Figura 2 mostra-se a série temporal formada pelas velas com seus preços máximos e mínimos.

Figura 2 – Preços e volume diários formado pelas velas da série temporal do Bitcoin.



Fonte: Elaborada pelo autor.

2.2 Hipótese do Mercado Eficiente

A Hipótese do Mercado Eficiente (EMH, do inglês *Efficient Market Hipothesis*) foi introduzida por Malkiel e Fama (1970) e sugere que o preço das ações reflete (em qualquer momento do tempo), imediatamente, toda a informação disponível pública e que a única causa segura da mudança do preço é nova informação (MALKIEL; FAMA, 1970). Um mercado que sempre reflete toda informação é chamado de "eficiente". Existem três formas diferentes dessa teoria, dependendo da natureza da informação: *weak-form efficiency*, *semi-strong form efficiency* e *strong-form efficiency*.

A *weak-form* afirma que os preços já refletem todas as informações públicas passadas disponíveis. Por enquanto, os preços futuros não podem ser previsíveis com base em análise de dados históricos. Isso faz com que os futuros movimentos dos preços são determinados por informações não contidas em preços passados.

A forma *semi-strong* declara que os preços mudam instantaneamente para refletir novas informações públicas. Ou seja, informações privadas não fazem parte.

A *strong form* afirma que grupos de investidores ou grupos exclusivos têm acesso exclusivo a qualquer tipo de informação da formação dos preços. A forma forte da EMH adicionalmente afirma que os preços refletem instantaneamente informações ocultas, privadas ou "internas".

A EMH está relacionada a duas abordagens de análise de investimentos. A Análise técnica e a Análise fundamental. A Análise Técnica declara que toda a informação já está incluída no preço e acredita que o futuro do preço pode ser previsto a partir do comportamento do passado histórico. A Análise Fundamental baseia as previsões do preço a partir de dados internos e externos da companhia como dados econômicos, parcerias e o valor dela (ARCE, 2017).

2.3 Análise do Movimento do Preço

Atualmente, existem duas técnicas para a análise do movimento do preço e a filosofia relacionada a cada uma delas. A análise Técnica e a análise Fundamental são duas abordagens usadas até hoje em dia.

Segundo Murphy (1999), "a Análise Técnica é o estudo do movimento da ação do mercado (do termo em inglês *market action*) através do uso de gráficos com o objetivo de prever o preço das tendências futuras do preço". A ação do mercado tem três fontes de informação: o preço e o volume. A Análise Técnica é uma disciplina amplamente complexa e distribuída que faz uso de ferramentas para análise do volume, preço.

Para Kirkpatrick e Dahlquist (2010), a Análise Técnica se baseia na tendência de mercados. Isto quer dizer que os participantes desejam comprar a preço barato no "início" de uma tendência e vender ao "final" de uma tendência. Embora isto pareça muito fácil, é na verdade muito complexo. O termo tendência será formalmente definido na seção 3.2.1. O mesmo autor declara que a Análise Técnica se baseia em:

a) Os preços das ações são determinados unicamente pela interação entre demanda e oferta.

b) Os preços das ações tendem a se mover em tendências.

c) Mudanças na demanda e na oferta causam reversões nas tendências.

d) Mudanças na demanda e oferta podem ser detectadas em gráficos.

e) Padrões gráficos tendem a se repetir.

Os defensores da Análise Técnica acreditam que "o mercado está sempre correto". Em outras palavras, não há espaço para considerar fatores externos que influenciam na demanda e na companhia. Esses fatores externos já estão incorporados no preço.

A Análise Técnica se apoia em indicadores ou ferramentas matemáticas. Os primeiros estudos nessa área datam desde 1960, com o *Rate of Change* (ROC) ou momentum. Na década de 1970, com o desenvolvimento dos gráficos por computador e velocidade para processar, apareceu um dos mais populares indicadores técnicos é o *Relative Strengh Index* (RSI) (KIRKPATRICK; DAHLQUIST, 2010).

Estes indicadores técnicos são importantes já que serão utilizados em diferentes partes de nossa proposta. O indicador RSI será utilizado na Seção 7.3.3. Os outros indicadores

utilizaremos como características (*input*) para extrair padrões úteis pelo modelo de extração de características compactas explicados nas Seções 7.4 e 8.3. As seguintes definições e fórmulas foram tomadas de Tulip[1]. Consideremos o preço de fechamento como P_t a t-ésima observação então o cálculo para os principais indicadores técnicos utilizados nesta pesquisa tem:

SMA Simple Moving Average: uma média móvel é calculada a partir dos preços médios de fechamento para um período especificado.

$$SMA_n = \frac{P_t - P_{t-1} + ... + P_{t-n}}{n} \tag{1}$$

MA Weighted Moving Average: atribua uma ponderação mais pesada a pontos de dados mais atuais, pois eles são mais relevantes do que os pontos de dados no passado distante.

$$WMA_n = \frac{P_t \times n + P_{t-1} \times (n-1) + ... + P_{t-n}}{\frac{n \times (n+1)}{2}} \tag{2}$$

EMA Exponencial Moving Average: também são ponderados em relação aos preços mais recentes, mas a taxa de redução entre um preço e seu preço anterior não é consistente. A diferença na diminuição é exponencial. A fórmula para EMA é:

$$EMA_n = P_t \times K + SMA_y \times (1-k) \tag{3}$$

Onde t é o período de hoje, $k = 2/(\text{observações num período} + 1)$, y é observação anterior.

RSI Relative Strange Index: é uma técnica usada para investir usando momentum e identificar ações de valor. O RSI fornece sinais aos traders técnicos sobre o momentum de alta e baixa dos preços e geralmente é plotado abaixo do gráfico do preço de um ativo. Um ativo é geralmente considerado sobrecomprado quando o RSI está acima de 70% e sobrevendido quando está abaixo de 30%.

$$RSI_t = 100 - \frac{100}{1 + (\frac{\text{average gain}}{\text{average loss}})} \tag{4}$$

$$RSI_t = 100 - \frac{100}{1 + \frac{sup_t}{sdown_t}} \tag{5}$$

onde:

$$up_t = \begin{cases} in_t - in_{t-1} & \text{if } in_t > in_{t-1} \\ 0 & \text{else} \end{cases}$$

[1] https://tulipindicators.org/

$$down_t = \begin{cases} in_{t-1} - in_t & \text{if } in_t < in_{t-1} \\ 0 & \text{otherwise} \end{cases}$$

$$sup_t = \frac{n-1}{n} sup_{t-1} + \frac{1}{n} up_t$$

$$sdown_t = \frac{n-1}{n} sdown_{t-1} + \frac{1}{n} down_t$$

Stochastic RSI Stochastic Relative Strange Index: O Stochastic RSI é um oscilador de momentum para ajudar a identificar tendências.

$$StochrRSI_t = \frac{rsi_t - min_t}{max_t - min_t} \tag{6}$$

onde:

$$max_t = maximum(rsi_t, rsi_{t-1}, ..., rsi_{t-n+1})$$
$$min_t = minimum(rsi_t, rsi_{t-1}, ..., rsi_{t-n+1})$$

MACD Moving Average Convergence Divergence : é um indicador de momentum de acompanhamento de tendências que mostra a relação entre duas médias móveis do preço de um título. O MACD é calculado subtraindo a média móvel exponencial de 26 períodos EMA da EMA de 12 períodos. A fórmula é a seguinte para o período p:

$$\begin{aligned} short_t &= EMA(12) \\ long_t &= EMA(26) \\ Macd_t &= short(12) - long(12) \\ signal &= EMA(p, Macd_t) \end{aligned} \tag{7}$$

CCI Commodity Channel Index: é usado para detectar tendências. Ele funciona tomando uma média móvel simples SMA do preço típico e comparando-a com a quantidade de volatilidade no preço típico. O CCI é um indicador de mercado usado para rastrear os movimentos do mercado que podem indicar compra ou venda.

$$\begin{aligned} typprice_t &= \frac{high_t + low_t + close_t}{3} \\ atp_t &= \frac{1}{n} \sum_{i=0}^{n-1} typprice_{t-i} \\ md_t &= \frac{1}{n} \sum_{i=0}^{n-1} |typprice_{t-i} - atp_t| \\ CCI_t &= \frac{typprice_t - atp_t}{0.015 md_t} \end{aligned} \tag{8}$$

BBANDS Bollinger Bands: é uma ferramenta de análise técnica desenvolvida para gerar sinais de sobrevenda ou sobrecompra. Projetado para descobrir oportunidades que dão aos investidores uma maior probabilidade de identificar corretamente quando um ativo está sobrevendido ou comprado em excesso. São três linhas que compõem as Bandas de Bollinger: Uma média móvel simples SMA (banda média) e uma banda superior e inferior. A fórmula é a seguinte:

$$
\begin{aligned}
bbands_t^{middle} &= \frac{1}{n}\sum_{i=0}^{n-1} in_{t-i} \\
bbands_t^{lower} &= bbands_t^{middle} - a\sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(in_{t-i} - bbands_t^{middle})^2} \\
bbands_t^{upper} &= bbands_t^{middle} + a\sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(in_{t-i} - bbands_t^{middle})^2}
\end{aligned}
\tag{9}
$$

A década de 1930 marcou o início da Análise Fundamental, com a criação da *Security Exchange Commision* (SEC), criada para regular o mercado e defendê-lo de manipulações. Formalmente, ela foi definida no ano 1934, com a publicação do livro *Security Analisys*, de Graham e Dodd (1934). A Análise Fundamental foi, então, desenvolvida durante grande parte do século XX. Hoje, a Análise Fundamental refere-se à negociação na expectativa de que os preços se movam para o nível previsto pelas relações entre a oferta e demanda, os fundamentos da teoria econômica (ALDRIDGE, 2009).

Análise Fundamental é uma técnica que tenta determinar os valores dos *assets* enfocando-se em fatores que afetam a companhia e seu futuro ou expectativa. A Análise Fundamental é composta pela análise da economia, análise da indústria e análise da empresa. Ao avaliar as condições gerais externas a empresa como a economia, o cenário da indústria que ela está, e como as informações da empresa foram publicamente disponíveis e acessíveis, o preço da ação foi determinado (CHRISTIE; ISIDORE, 2018).

Essa análise responde a perguntas como: a empresa é capaz de pagar suas dívidas? A empresa tem crescimento futuro? Atualmente está fazendo dinheiro?. A empresa é realmente boa para investir?. Esta análise, analisa, as condições econômicas e o valor que uma companhia terá no longo prazo. A Análise Fundamental e a Análise Técnica coexistem e foram usadas como as principais ferramentas para a tomada de decisões na hora de realizar transações no mercado de ações locais e globais. Dependendo dos investidores, existem alguns que preferem utilizar alguma delas ou ambas ferramentas combinadas.

Capítulo 3

Previsão de Séries Temporais

Neste capítulo apresenta-se os conceitos relacionados à previsão de series temporais como um campo importante de estudos do *forecasting*, onde observações passadas são analisadas para desenvolver um modelo que descreve as relações internas das variáveis observadas.

3.1 Previsão

De acordo a Hyndman e Athanasopoulos (2018), a tarefa de previsão (*forecasting*) é uma tarefa estatística comum nos negócios, que ajuda a acrescentar informação às decisões sobre o cronograma de produção, demanda de transporte, consumo energético, entre outros e fornece uma guia para o planejamento estratégico de longo prazo. A previsão é uma ajuda importante no planejamento eficaz e eficiente, existem algumas tarefas que são mais fáceis que outras. Por exemplo, taxas de câmbio são muito difíceis de prever com precisão.

Em outra definição, encontra-se em Montgomery, Jennings e Kulahci (2015) que o *forecasting* é uma tarefa de previsão de algum evento ou eventos. A previsão pode ser *short-term*, o que refere-se a predizer períodos curtos de tempo (horas, dias, semanas), *mid-term*, que pode ser um ou dois anos, ou *long-term*, podendo se estender a vários anos. É uma tarefa muito importante em muitos campos como: na indústria, no governo, na economia, na medicina, na política e nas finanças.

Os métodos de previsão podem ser classificados em duas categorias: métodos qualitativos e métodos quantitativos (HYNDMAN; ATHANASOPOULOS, 2018), conforme detalhado a seguir.

3.1.1 Previsão Quantitativa

A previsão quantitativa baseia seu estudo apenas no histórico para realizar as futuras previsões. Existem duas condições para que se apliquem: assumir que alguns padrões se repetirão no futuro e deve existir uma base histórica de dados. Essa categoria trabalha com variadas técnicas de previsão (que serão aprofundadas mais afrente), cada método tem suas próprias características, vantagens e desvantagens.

Segundo Hyndman e Athanasopoulos (2018), a previsão quantitativa se divide em:

1. **Modelos explicativos**, que assumem que a variável a prever têm algum tipo de relação explicativa com uma ou mais variáveis. Esses modelos são úteis porque não precisam de valores históricos das variáveis a prever. Por exemplo, a Demanda de Eletricidade (DE) de uma região em verão, pode ser definida por:

   $$DE = \text{f(Current Temperature , Strength Economy, Population, Time of Day, Time of Week, erro)}$$

   Tais relações não são exatas. O erro na fórmula permite uma variação aleatória e efeito nas variáveis não incluídas no modelo. O propósito que tem um modelo explicativo é descrever a forma do relacionamento e o uso para prever os valores futuros da variável de previsão.

2. **Modelos de séries temporais** usam só informação da variável a prever e não pretendem descobrir os fatores que afetam seu comportamento.

   $$DE_t = f(DE_{t-1}, DE_{t-2}, DE_{t-3}, ..., erro)$$

   Onde t é a presente observação, $t-1$ é a observação anterior e assim por diante. Aqui, a previsão do futuro é baseada em valores passados por variáveis ou erros passados. Nesta categoria, existem os modelos de séries temporais para a previsão chamados *ARIMA*, *Exponential Smoothing* entre outros (PARMEZAN; SOUZA; BATISTA, 2019).

3. **Modelos mistos** combinam características dos modelos mencionados anteriormente.

   $$DE_{t+1} = f(DE_t, \text{Strength Economy, Population, Time of Day, Time of Week, erro})$$

   Esses tipos de modelos têm diferentes nomes em diferentes disciplinas, quais são conhecidos como Modelos de Regressão Linear Dinâmicos e modelos lineares(onde assume que f é linear).

Para os autores Hyndman e Athanasopoulos (2018) consideram que utilizar um modelo de série temporal sem informação explicativa ou mista é mas preciso na hora de prever e não recomenda seu uso.

3.1.2 Previsão Qualitativa

Esses métodos são usados quando **não existêm dados históricos** para fazer a previsão, por isso são chamados como previsão por julgamentos (HYNDMAN; ATHANASOPOULOS, 2018). Por exemplo, a previsão qualitativa pode ser aplicada quando um produto vai ser lançado e não possui dados históricos e só depois de um tempo haverá dados históricos acumulados.

Segundo Hyndman e Athanasopoulos (2018), pesquisas nesta área mostraram que a precisão da previsão de julgamento melhora quando o previsor tem: (i) conhecimento importante de domínio e (ii) uso de informações mais atualizadas e oportunas. Também servem para melhorar a previsão quantitativa, levando em conta informações que não podem ser incorporadas em um modelo formal e puramente estatístico. Mas, para o autores, as previsões estatísticas são geralmente superiores à geração de previsões usando apenas o julgamento.

Existem três configurações gerais nas quais a previsão de julgamento é usada Hyndman e Athanasopoulos (2018):

a) Não há dados históricos então os métodos estatísticos não são aplicável. É necessária a previsão baseada no julgamento subjetivo.

b) Quando os dados estão disponíveis, então deve fazer as previsões estatísticas, e depois ser ajustadas usando o julgamento.

c) Existem dados históricos, fazer as previsões estatísticas, fazer as previsões baseados no julgamento independentemente e depois combinadas.

3.2 Séries Temporais Financeiras

Uma série temporal é uma sequência de observações (discretas ou contínuas) de uma variável de inteiros, ordenadas cronologicamente $X_1, X_2, X_3...X_n$ (MONTGOMERY; JENNINGS; KULAHCI, 2015). É importante destacar que as séries possuem uma dependência de ordem e mudar essa ordem modifica o significado do dado.

Outra definição encontra-se em Parmezan, Souza e Batista (2019), onde os autores definem uma série temporal como uma série de observações ordenadas Z de tamanho t, $Z = (z_1, z_2, z_3...z_t)$ onde $z_t \in \mathbb{R}$ para todos os $t \in [1, m]$. Cada valor z_t de Z é uma observação da série temporal no instante t. Se os valores da série são sintetizados por uma função matemática $y = f(time)$ a série chama-se determinística e quando a série adiciona um termo aleatório ϵ: $y = f(time, \epsilon)$ a série é estocástica ou não determinística.

Adiante, se definirá uma característica relevante das séries temporais, a "estacionariedade". Uma série é estacionária se desenvolve-se aleatoriamente em torno de uma média constante, que reflete algum equilíbrio estável. Essa propriedade é essencial, pois diversos

métodos assumem tal condição para prever os valores futuros (PARMEZAN; SOUZA; BATISTA, 2019).

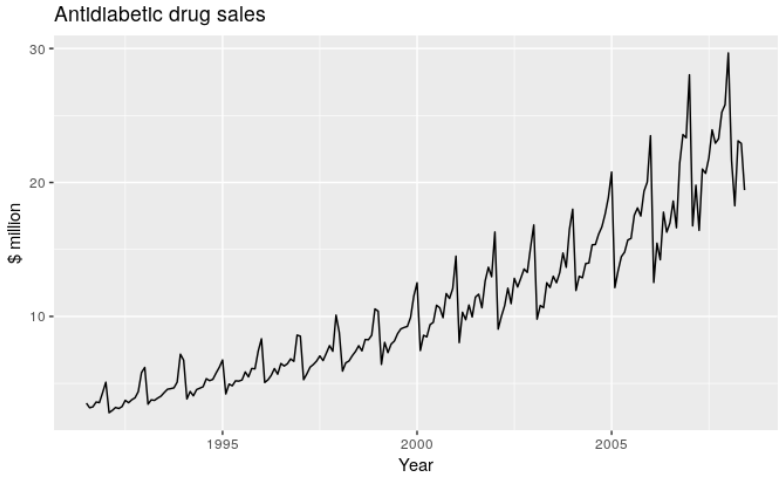
Segundo Adhikari e Agrawal (2013), as séries com uma variável são chamadas *univariadas* e quando têm mais de uma variável são chamadas *multivariadas*. Além disso, elas podem ser contínuas ou discretas. As contínuas podem ser medidas em qualquer tempo, no entanto as discretas normalmente têm observações em pontos discretos. Usualmente as séries com tempo discretos são registradas em intervalos de tempo iguais (a cada milissegundo, a cada uma hora, diariamente, semanalmente, etc).

3.2.1 Componentes das Séries Temporais

As séries temporais são afetadas por três componentes Hyndman e Athanasopoulos (2018): a *tendência*, a *sazonalidade* e o *ciclo*.

1. **Tendência.** Existe uma tendência quando há um aumento ou uma diminuição a longo prazo dos dados. Não precisa ser linear. Às vezes nos referimos a uma tendência como uma "mudança de direção", podendo ser uma tendência crescente ou decrescente. Por exemplo, na Figura 3 e Figura 4 (esquerda-abaixo), pode-se observar o crescimento da tendência de vendas de antibióticos e a produção de eletricidade, respectivamente, com forte incremento. Nesse caso, de acordo com Bhattacharyya (1984), pode-se dizer que se $X_1 < X_2 < \ldots < X_n$ há uma tendência em qualquer janela de tempo.

Figura 3 – Tendência crescente de vendas desde $1995 - 2009$ de remédio.



Fonte: Hyndman e Athanasopoulos (2018).

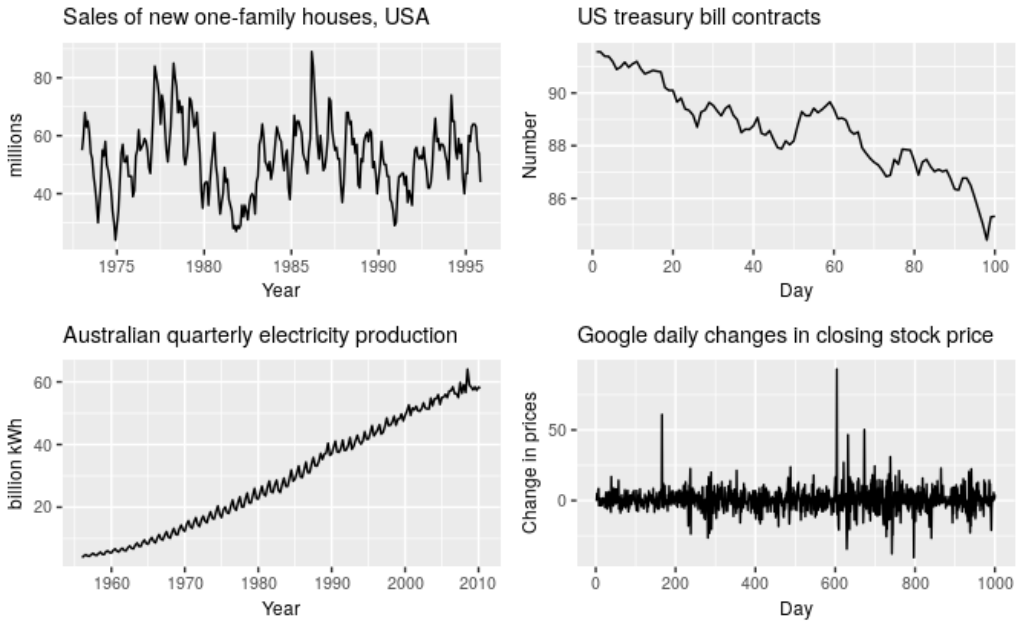
2. **Sazonalidade.** Um padrão sazonal ocorre quando uma série temporal é afetada por fatores sazonais, como a época do ano, o dia da semana, as condições do clima, ou hábitos tradicionais. A sazonalidade é sempre de uma frequência fixa e conhecida. Por exemplo: as vendas mensais de medicamentos antidiabéticos, ilustradas na

Figura anterior 3, ilustra a sazonalidade, que é induzida em parte pela mudança no custo dos medicamentos no final de cada ano. Outro exemplo seria as vendas de sorvete quando o clima está quente e as vendas de roupas quentes no inverno.

3. **Ciclo**. Ocorre quando as séries exibem subidas e caídas que não são de frequência fixa. Essas flutuações são geralmente devidas a condições econômicas e estão frequentemente relacionadas ao "ciclo de negócios". Na Figura 4(esquerda-acima) mostra-se a combinação de uma tendência parte de um ciclo cumprido.

   Para Adhikari e Agrawal (2013), "a variação cíclica em uma série temporal descreve as mudanças de médio prazo na série, causadas por circunstâncias, que se repetem em ciclos. Normalmente a duração de um ciclo se estende por um longo período de tempo, geralmente dois ou mais anos".

Figura 4 – Combinações diferentes de padrões.



Fonte: Hyndman e Athanasopoulos (2018).

Na Figura 4 (Direita abaixo) pode-se observar os valores de fechamento de *Google* sem nenhum dos componentes antes mencionados tendência, Sazonalidade, ciclo.

Muitas séries temporais incluem esses três tipos de padrões. Para Hyndman e Athanasopoulos (2018) é muito importante identificar que **tipo de padrão há na série temporal** para poder escolher o método correto do modelo.

Alguns autores recomendam dividir estas séries em seus componentes para fazer algum tipo de análise por exemplo em Adhikari e Agrawal (2013) menciona que uma série pode ser dividida em componentes aditivos e multiplicativos assim:

$$\text{Multiplicativo:} \quad Y(t) = T(t) \times S(t) \times C(t) \times I(t)$$

$$\text{Aditivo:} \quad Y(t) = T(t) + S(t) + C(t) + I(t)$$

$T(t), S(t), C(t)$ refere-se a tendência, a sazonalidade e o ciclo. O valor $I(t)$ é um padrão irregular. Quando é multiplicativo os valores dependem entre sim e quando não têm dependência usa-se aditivos.

3.2.2 Características das séries Temporais

Existem diferentes características das séries temporais financeiras, entre as mais importantes temos a dependência temporal, a distribuição e a não linearidade.

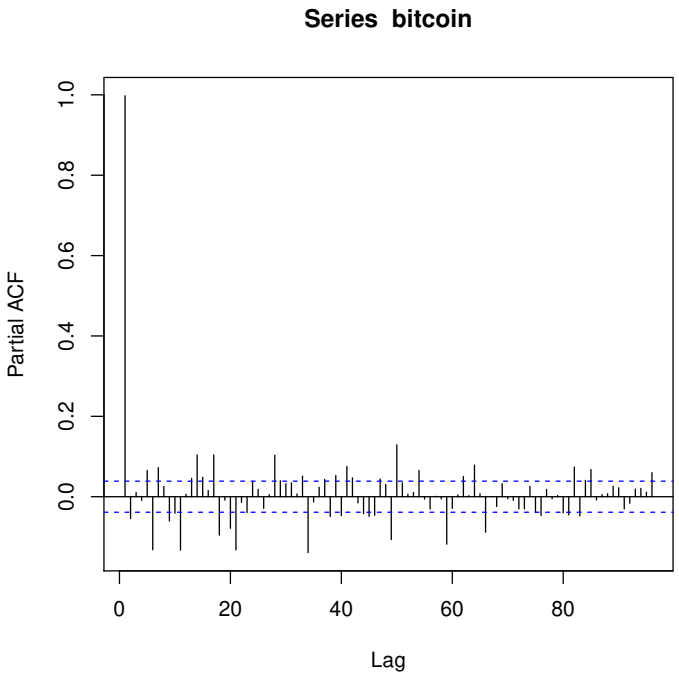
1. **Dependência**, que refere-se primeiro a correlação. A correlação mede a extensão de uma relação linear entre duas variáveis. Por outro lado a autocorrelação mede a relação linear entre valores defasados de uma série temporal Hyndman e Athanasopoulos (2018).

   Existem vários índices de autocorrelação por exemplo os retornos de uma série temporal y_t é definida como $r_t = y_t - y_{t-1}$. A *Auto Correlation Function* (ACF) que mede a previsibilidade linear da série temporal y_t no tempo t. Por exemplo r_1 mede a relação entre y_t e y_{t-1}, r_2 mede a relação entre y_t e y_{t-2} assim adiante. O ACF esta definido pela Equação 10:

   $$r_k = \frac{\sum_{t=k+1}^{T}(y_t - \hat{y})(y_{t-k} - \hat{y})}{\sum_{t=k+1}^{T}(y_t - \hat{y})^2} \tag{10}$$

   Onde T é a longitude da série temporal. Uma representação do ACF mostra-se na Figura 5.

Figura 5 – Correlograma ACF.



Fonte: Elaborada pelo autor.

2. **Não linearidade**, ocorre quando as séries temporais são não lineares em média ou não lineares em variância.

3. **Estacionariedade**, de acordo com Arce (2017), uma série estacionaria estritamente é uma série onde seu comportamento probabilísticos é constante para cada coleção de observações $\{y_1, y_2, ...y_t\}$ é idêntica nesse tempo. Formalmente:

$$P\{y_{t_1} < c_1, y_2 < c_2, ..., y_{t_L} < C_L\} = P\{y_{t_1+h} < c_1, y_2 < c_2, ..., y_{t_L+h} < C_L\},$$
$$\forall L \in \mathbb{N}, \forall h \in \mathbb{Z}$$

onde c_1, C_L são constantes. Esta definição é muito forte e existe outra definição mais débil. Uma série temporal estacionaria débil é quando a média, a variância e a auto-covariância não mudam no tempo. Veja as Equações:

$$E(y_t) = \mu \quad \forall t \in \mathbb{N} \tag{11}$$
$$E(y_t^2) = \sigma^2 \quad \forall t \in \mathbb{N} \tag{12}$$
$$\lambda(s,t) = \lambda(s+h, t+h) \quad \forall s, t \in \mathbb{N}, \forall h \in \mathbb{Z} \tag{13}$$

onde $\lambda(s,t) = E[(y_s - \mu)(y_t - \mu)]$.

Outra definição encontra-se em Hyndman e Athanasopoulos (2018) onde, define que uma série temporal estacionária é aquela cujas propriedades não dependem do tempo em que a série é observada.

Assim, as séries temporais com tendências, ou com a sazonalidade, não são estacionárias já que a tendência e a sazonalidade afetarão o valor da série temporal em tempos diferentes. Em outras palavras uma série estacionária não tem padrões previsíveis a longo prazo.

4. **Processos Não Estacionários**, a autora Arce (2017) menciona que existem vários tipos de processos não estacionários encontrados na economia.

   a) **Tendência determinística:** ou processo estacionário com tendência tem a seguinte forma:

   $$y_t = f(t) + \epsilon_t$$

   onde t é o tempo e ϵ_t representa um termo de erro estacionário (com média 0 e variância σ^2). E $f(t)$ é uma função determinística do tempo. Se $f(t)$ é:

   - ❏ Se $f(t) = \alpha + \beta t$, é um modelo com tendência linear.
   - ❏ Se $f(t) = \alpha \cdot expre^r t$, é uma curva com crescimento exponencial
   - ❏ Se $f(t) = c_1 + c_2 t + c_3 t^2$, é um modelo com tendência quadrática.
   - ❏ Se $f(t) = \frac{1}{k+\alpha\beta^t}$, é uma curva logística.

   b) **Tendência Estocástica:** têm a seguinte forma:

   $$y_t = \mu + y_{t-1} + \epsilon_t$$

   onde ϵ_t representa um processo estacionário. Quando $\mu = 0$ o processo é chamado puramente *Random Walk* e quando $\mu \neq 0$ é um processo chamado *Random Walk* com desvio.

Uma série temporal é uma sequência de observações (discretas ou contínuas) de uma variável de inteiros, ordenadas cronologicamente $X_1, X_2, X_3 ... X_n$ (MONTGOMERY; JENNINGS; KULAHCI, 2015). É importante destacar que as séries possuem uma dependência de ordem e mudar essa ordem modifica o significado do dado.

Capítulo 4

Modelos de Aprendizado de Máquina

O aprendizado de máquina é uma disciplina focada no desenvolvimento de algoritmos que sejam capazes de aprender a partir dos dados de experiência. Esta ideia é o foco central do desenvolvimento de algoritmos como estratégias de robôs, processamento de língua natural, previsão de séries financeiras, entre outras tarefas.

Os modelos apresentados servem para modelar e prever variáveis dependentes nas séries temporais apresentadas no capítulo anterior.

4.1 Aprendizado de Máquina

O Aprendizado de Máquina (AM) é uma área de pesquisa cujos objetivos gerais são o desenvolvimento de técnicas computacionais que permitem simular o processo de aprendizado e a construção de sistemas capazes de adquirir conhecimento de maneira automática (MITCHELL, 1997). Nestas seções vão apresentar-se os conceitos de AM que serão utilizados nesta pesquisa.

4.1.1 Algoritmos de Aprendizado

Um algoritmo de aprendizado é um algoritmo que é capaz de aprender algum conceito ou padrão a partir de dados. Uma definição exata encontra-se em Mitchell (1997) que diz: "um programa de computador aprende a partir da experiência **E** com relação a alguma classe de tarefas **T** e medida de desempenho **P**, se seu desempenho das tarefas **T**, são medidas com **P**, melhora com a experiência **E**".

O termo **T** refere-se à habilidade de realizar uma tarefa específica. Por exemplo, se quer dar a um robô a habilidade de nadar, então aprender a nadar seria a tarefa. O termo

P é a medida de desempenho da habilidade **T**. O termo **E** indica a experiência, que é apresentada como o conjunto de *dataset* ou uma coleção de muitos pontos.

A tarefa **T** pode ser uns dos tipos de tarefas como a regressão, a classificação, o agrupamento etc. Para uma melhor compreensão serão apresentados os tipos de aprendizado, especialmente enfocando-se no Aprendizado Supervisionado.

4.1.2 Tipos de Aprendizado

Nesta seção se explicará os tipos de aprendizado sobre todo um tipo de aprendizado guiado por um supervisor e não se aprofundara em outros tipos de aprendizado que existem.

Os Algoritmo de AM podem ser organizados de acordo a diferentes critérios. Uns desses critérios diz respeito ao paradigma de aprendizado adotado para resolver a tarefa. Segundo Faceli et al. (2000), as tarefas podem ser divididas em "preditivas" e "descritivas".

1. **Aprendizado Supervisionado** Os algoritmos que induzem modelos preditivos seguem o paradigma de Aprendizado Supervisionado. Ao falar-se de supervisão, refere-se ao fato de se assumir a presença de um supervisor que conhece a saída (ou "rótulo") esperada para cada dado Faceli et al. (2000).

   Os algoritmos supervisionados são métodos que procuram descobrir a relação entre atributos (variáveis independentes) e um atributo objetivo (variável dependente). A relação descoberta é representada em uma estrutura referida como um *modelo*. Esses modelos descrevem e explicam os fenômenos que estão escondidos no conjunto de dados e podem ser usados para predizer o valor do atributo alvo, sabendo os valores dos atributos de entrada (ROKACH, 2010).

   Para Arce (2017), o aprendizado supervisionado vem do supervisor que atua como um professor no processo de aprendizado, onde o objetivo é aprender uma regra geral que faz o mapeamento das entradas a saídas otimizadas com uma função.

   Existem dois tipos de aprendizado supervisionado: a classificação e a regressão. Os modelos de regressão mapeiam o espaço de entrada a um valor real. Os modelos de classificação mapeiam a entrada a um espaço de classes pré definidas.

2. **Aprendizado Semissupervisionado** Quando construímos classificadores para tarefas específicas, induzimos que há um conjunto de dados com todos os exemplos rotulados.

   No mundo real, existem situações onde há muitos dados não rotulados ou se precisa conhecimento do domínio de aplicação para rotular como: a indexação de vídeo, diagnósticos médicos, modelagem de tópicos entre outros. Além disso, a rotulação de dados leva tempo e pode consumir recursos caros, muitas vezes não disponíveis.

A grande motivação deste tipo de aprendizado se dá pela abundância de exemplos não rotulados e a ausência de exemplos rotulados. Para esses casos, se utiliza um conjunto menor de dados corretamente rotulados para espalhar o rótulo a dados que não possuem rótulo.

Uma definição mais exata encontra-se em (BRUCE, 2001), que menciona que a ideia central deste tipo de aprendizado semissupervisionado é utilizar os exemplos rotulados para então obter informações sobre o problema e utilizá-las para guiar o aprendizado a partir dos exemplos não rotulados. Vários algoritmos têm sido propostos, nas próximas seções são descritos mais detalhadamente.

3. **Aprendizado Não Supervisionado** Em tarefas de descrição, o alvo é descrever um conjunto de dados de forma a enriquecer o seu entendimento. Os algoritmos de AM utilizados neste grupo não fazem uso de rótulos. Por exemplo, o *clustering* é uma tarefa que busca agrupar dados semelhantes baseadas em centroides. Outra tarefa descritiva é encontrar regras de associação, que relacionem grupos de atributos entre outros grupos Faceli et al. (2000).

   Neste tipo de aprendizado sem supervisor, também existem as tarefas de redução da dimensionalidade e aprendizagem de representações dos dados sem necessidade do rótulo, eles conseguem dividir grupos com características similares (GOODFELLOW; BENGIO; COURVILLE, 2016). Na Secção 4.3 será explicado os *Autoencoders* como um tipo de aprendizado não supervisionado.

Algumas técnicas de Aprendizado de Máquina podem ser vistas como problemas de otimização. Este processo consiste em definir um critério de aprendizagem, ou seja, a função a ser otimizada em que o objetivo é maximizar ou minimizar uma função objetivo (FACELI et al., 2000). Existem muitos exemplos de problemas de aprendizado de máquina: detecção de anomalias, previsão de séries temporais, tradução de máquina, classificação, processamento de imagem, detecção de rosto, *spam filtering*, previsão do tempo, entre muitos outros.

4.2 Redes Neurais Artificiais

Pela constante busca de uma máquina inteligente, o modelo que aparece naturalmente é o cérebro humano. Em nosso dia a dia, realizamos diferentes tarefas como caminhar, falar, olhar etc. Todas essas tarefas requerem de nós a atenção a diferentes eventos ao mesmo tempo e o processamento de diferentes informações. Essas tarefas requerem processos simples como a memória, processamento em paralelo, etc.

Todas estas tarefas é possível a nossa complexa estrutura biológica o cérebro humano que esta a cargo de gerar respostas a estímulos externos. O cérebro humano é parte de

nosso sistema nervoso, é um conjunto complexo de células que determinam o funcionamento dos seres vivos, a unidade fundamental do sistema nervoso é a célula nervosa, o neurônio que se distingue das outras células por apresentar excitabilidade que lhe permite responder a estímulos externos e internos (FACELI et al., 2000).

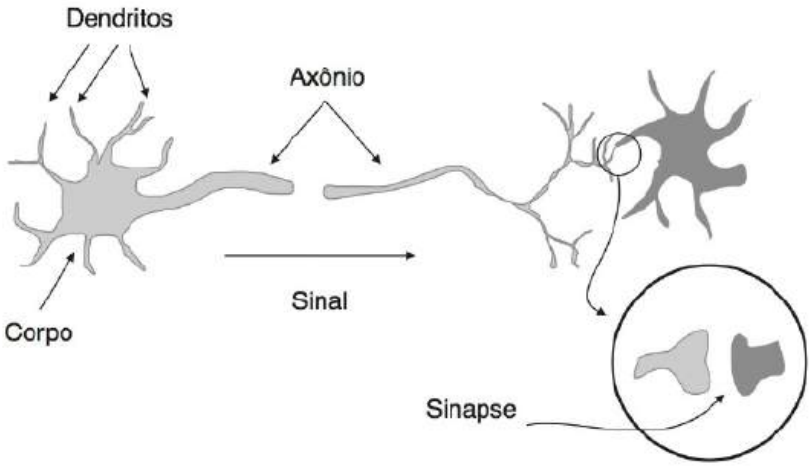
O principal bloco de construção do cérebro é o neurônio, os principais componentes são: dendritos, corpo celular e axônio veja a Figura 6.

a) **Dendritos**, são as terminações dos neurônios e encargados da recepção da informação proveniente do ambiente.

b) **Corpo celular**, a informação é transmitida ao corpo celular ou soma, onde é combinada e processada a informação.

c) **Axônio**, é um prolongamento dos neurônios responsável pela condução dos impulsos elétricos produzidos no corpo celular

Figura 6 – Neurônio biológico com suas partes.



Fonte: Faceli et al. (2000).

Denomina-se por sinapse o contato o dendrito de um neurônio e o axônio de outro neurônio. As sinapses são estruturas fundamentais que interagem com outros neurônios. A sinapse mais comum é a química, que converte o sinal elétrico pre-sináptico em um sinal químico e retorna um sinal pós-sináptico. Se estima que o cérebro tenha entre 10 bilhões de neurônios e 60 trilhões de sinapse ou conexões (HAYKIN, 1994).

Na literatura existem várias definições para as Redes Neurais Artificiais (RNA) (do inglês *Artificial Neural Network*, ANN). As ANN podem ser definidas como sistemas computacionais que processam informações baseando-se no funcionamento interno do cérebro entre neurônios e suas conexões com outros neurônios. Em outras palavras, elas foram inspiradas na estrutura biológica do cérebro, com um número grande de unidades de processamento, chamadas neurônios, organizadas em camadas, com a capacidade de aprender alguma tarefa (por exemplo classificação) ajustando os pesos entre os neurônios.

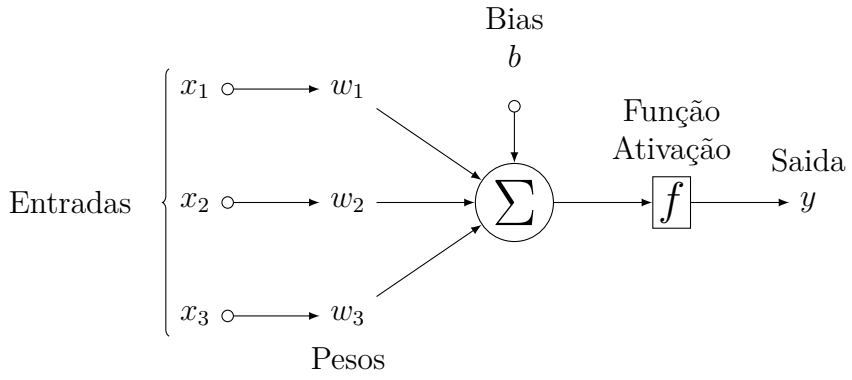
Outra definição para ANN menciona que:

"Rede Neural Artificial é definida como um processador distribuído paralelo composto de unidades de processamento simples, que possui uma propensão natural para armazenar conhecimento experimental e torná lo disponível para utilização. O conhecimento é adquirido pela rede a partir de seu ambiente, por meio de um processo de aprendizado, e as forças das conexões entre os neurônios artificiais da rede, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido." Cerri (2013 apud HAYKIN, 1994, p. 15-16)

Para Faceli et al. (2000) uma ANN é categorizada por dois aspectos: a *arquitetura* e o *aprendizado*.

A arquitetura, refere-se ao tipo e número de unidades de processamento e a forma como os neurônios estão conectados. O neurônio é a unidade fundamental de uma ANN, como apresentado na Figura 7, onde cada entrada de dados x do neurônio simula os dendritos, cujos valores são ponderados e somados por uma função **f**.

Figura 7 – Neurônio simples.



Fonte: Elaborada pelo autor.

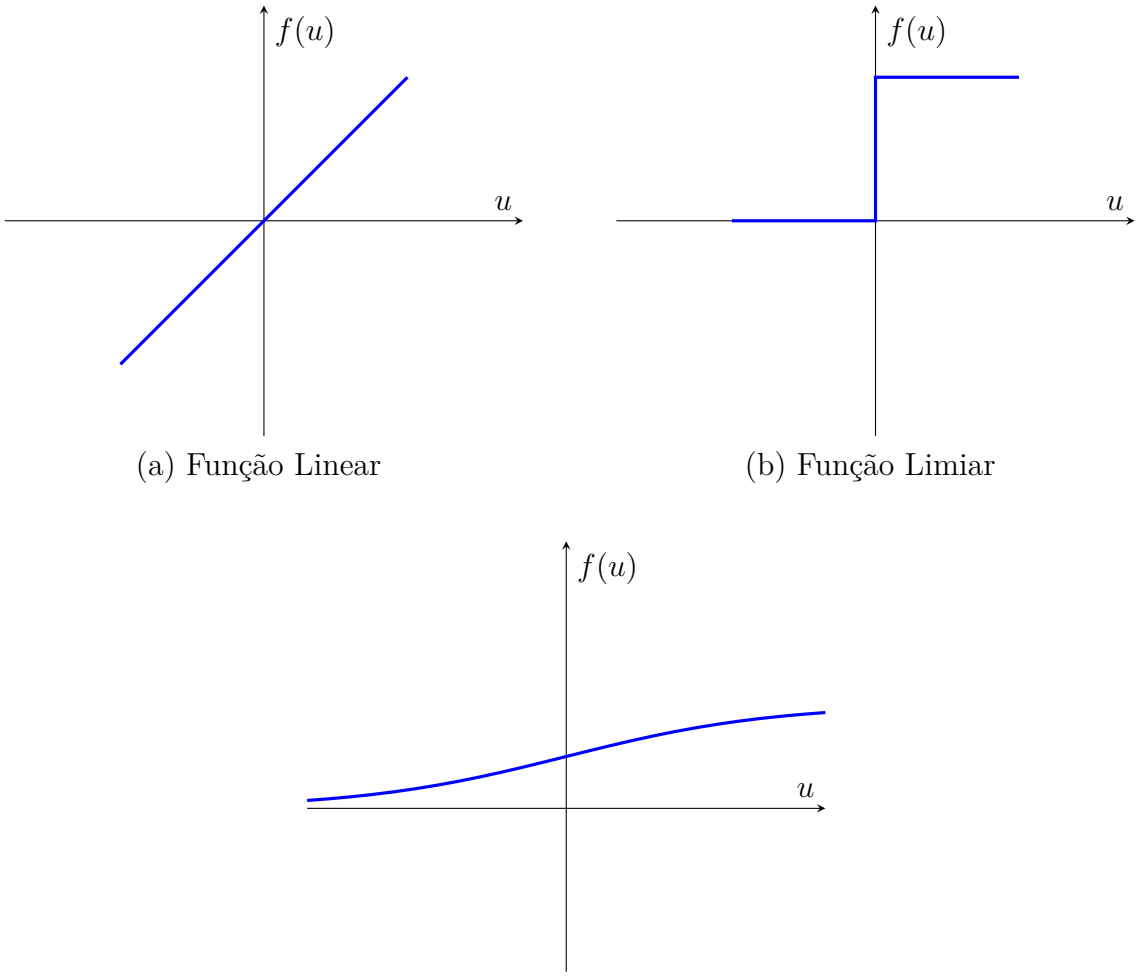
A saída da função é a resposta do neurônio para uma próxima entrada. A definição matemática do neurônio **u** está definida pela Equação 14 (FACELI et al., 2000):

$$\mathbf{u} = \sum_{j=1}^{d} x_j * w_j \tag{14}$$

onde x é um objeto com d atributos $x = [x_1, x_2, x_3...]$ e o neurônio têm d terminais de entrada com pesos: $w = [w_1, w_2, w_3, ..., w_d]$.

A saída de um neurônio é definida por meio da aplicação de uma função de ativação sobre a entrada total. Na literatura, foram propostas muitas funções de ativação, como linear, limiar (ou degral) e sigmoidal, ilustradas na Figura 8. A função linear na Figura 8(a) retorna como saída o valor de u, a função limiar na Figura 8(b) define quando o resultado será igual a 1 ou 0. A função sigmoidal na Figura 8(c) representa uma aproximação contínua e diferenciável da função limiar.

Figura 8 – Funções de Ativação adaptadas.



(a) Função Linear (b) Função Limiar

(c) Função Sigmoidal

Fonte: Elaborada pelo autor.

Em uma ANN, os neurônios podem se estruturar em camadas. Uma rede com mais de uma camada de neurônios recebe o nome de rede multicamadas. Nesse caso, a camada de neurônios que gera os valores de saída é chamada camada de saída, e as demais camadas são chamadas camadas ocultas.

As ANN podem apresentar ou não conexões de retroalimentação ou *feedback*. A informação de uma rede neural flui desde a camada de entrada até a camada de saída. Nas redes multicamadas, esse fluxo ocorre de camada em camada. As conexões de retroalimentação permitem que um neurônio receba em seus terminais de entrada a saída de um neurônio da mesma camada ou de camadas posteriores. O neurônio pode inclusive receber sua própria saída em um de seus terminais de entrada. Essas são conhecidas como redes recorrentes, que são úteis para a modelagem de sequências (FACELI et al., 2000). Para Zhang, Wang e Liu (2018), essas redes podem estar categorizadas pela topologia em: *Feedfordward Neural Network* (FNN) e *Recurrent Neural Network* (RNN). Finalmente, o número de neurônios, o número de camadas e a presença de conexões de retro-propagação definem a topologia de uma rede ANN. O aprendizado, diz respeito às regras utilizadas

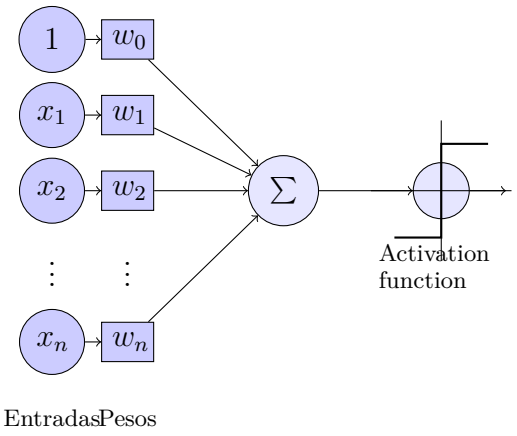
para o ajuste dos pesos da rede. O ajuste é a definição de valores dos pesos associados as conexões da rede que fazem que o modelo obtenha melhor acurácia preditiva. Essas regras agrupadas são conhecidas como algoritmos de treinamento.

4.2.1 Perceptron

O perceptron é uma das formas mais básicas de uma ANN. O perceptron foi o primeiro modelo de neurônio artificial que utiliza o aprendizado supervisionado, desenvolvido por (ROSENBLATT, 1958). Essa rede utiliza o modelo de neurônio proposto por McCulloch e Pitts (MCCULLOCH; PITTS, 1943). A arquitetura é constituída somente de uma camada de um neurônio, que recebe estímulos do ambiente (dados de entrada) e os utiliza para construir um hiperplano capaz de separar dados em duas classes linearmente separáveis.

Na Figura 9, é ilustrada a arquitetura do perceptron com um neurônio. A cada entrada é associado um peso, que pode ser positivo ($w_i > 0$) ou negativo ($w_i < 0$). Para cada exemplo, os pesos são multiplicados pelo valor do atributo associado e somados, este valor somado é à sinapse do neurônio. A combinação das entradas com esse peso mais uma limiar (*bias*) produz a saída. Esse valor da saída é enviado a uma função de ativação, que define a saída y do neurônio.

Figura 9 – Perceptron básico adaptado.



Fonte: Elaborada pelo autor.

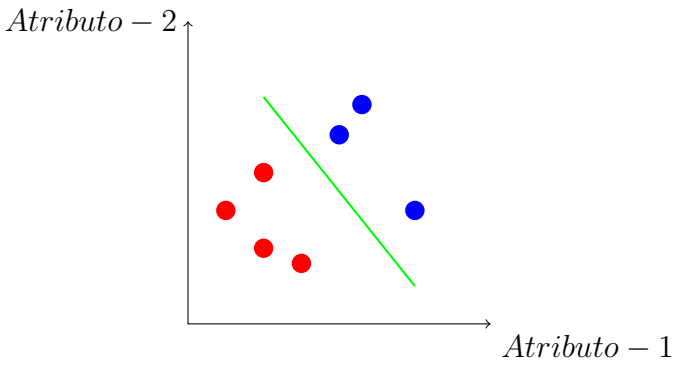
Segundo Faceli et al. (2000), os pesos sinápticos do perceptron são treinados com um algoritmo de *correção de erro* que usa um número finito de iterações e usa uma função de ativação do tipo limiar. Na fase de treinamento, os pesos são ajustado acordo a Equação 15:

$$w(t+1) = w(t) + \gamma w_i(y_i - f(x_i)) \tag{15}$$

Onde $w(t)$ é o peso da i-ésima conexão de entrada no instante t, γ é a taxa de aprendizado, x_i é o valor do i-ésimo atributo da entrada, $f(x_i)$ é a saída produzida pela rede e y_i é a saída desejada ou o rótulo para x_i.

A proposta do perceptron causou muito impacto nas pesquisas de Redes Neurais em sua época. Pouco mais tarde, as pesquisas foram interrompidas porque se demostrou que elas apenas conseguiam separar problemas linearmente separáveis, o que pode ser uma limitação muito relevante. Um problema separável é definido por um problema de classificação com duas classes (classificação binária) em que os objetos das duas classes podem ser separados por um hiperplano, conforme ilustrado na Figura 10.

Figura 10 – Objetos linearmente separáveis.



Fonte: Elaborada pelo autor.

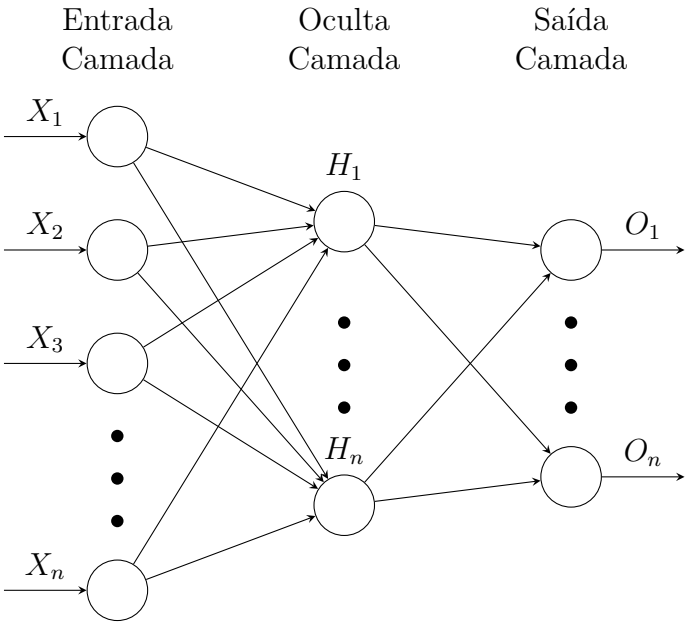
Para contornar esse problema, avanços levaram à criação de estruturas mais complexas que sejam capazes de utilizar uma ou mais camadas intermediárias, como a *Multi-Layer Perceptron*.

4.2.2 Multi-Layer Perceptron

O *Multilayer Perceptron* (MLP) é um tipo de rede neural que apresenta uma ou mais camadas intermediárias de neurônios, mais uma camada de saída. Esse tipo de rede é denominado como *Feed-Forward*, devido à propagação de sinapses em direção à camada de saída.

Na arquitetura de uma MLP, cada camada é representada com l e está completamente conectada à camada seguinte $l + 1$, ou seja, a saída de cada neurônio em uma camada intermediária compõe parte da entrada de todos os neurônios da camada seguinte. A entrada da primeira camada se refere a todos os atributos da entrada $X_i, i = (1, 2, 3 \ldots n)$ para um exemplo X, um para cada neurônio (FACELI et al., 2000). A Figura 11 ilustra este tipo de redes, veja:

Figura 11 – Multi-layer Perceptron básico.



Fonte: Adaptado de Haykin (1994)

As camadas intermediárias ou ocultas H_n têm a função de ser extratoras de características estatísticas de alto ordem (especialmente em aplicações com grande volume de dados e atributos), elas não interagem diretamente com o ambiente e a camada de saída O_n (HAYKIN, 1994).

Em uma MLP cada neurônio realiza uma função específica. A função de um neurônio de camadas intermediarias é uma combinação das funções realizadas pelos neurônios de camada anterior que estão conectadas a ele. A medida que o processamento avança de uma camada intermediaria à camada seguinte $l + 1$, o processamento realizado se torna mais complexo. Na primeira camada, cada neurônio aprende uma função que define o hiperplano, o qual divide o espaço de entrada em duas partes e os neurônios de camadas seguintes combina o grupo de hiperplanos para formar regiões convexas (FACELI et al., 2000).

Cada neurônio da camada de saída está associado a uma classe do conjunto de dados. Assim, os valores gerados pelos neurônios da camada de saída podem ser representados por um vetor $y = [y_1, y_2, \ldots, y_k]$, onde k é o número de classes do problema e, consequentemente, de neurônios da última camada (FACELI et al., 2000).

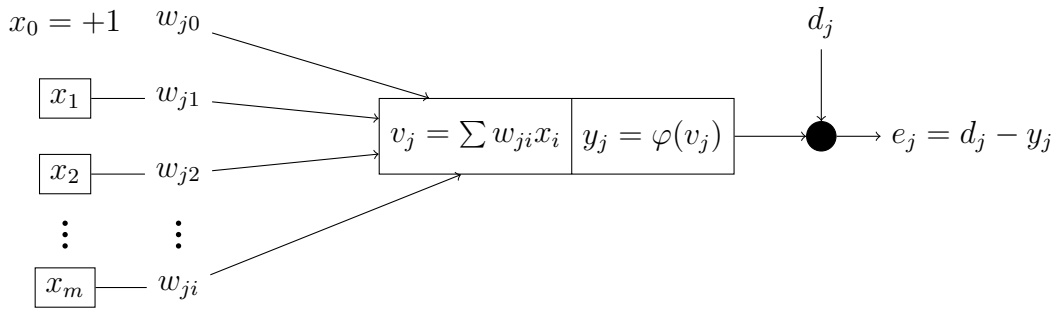
4.2.2.1 Algoritmo Back-Propagation

Uma das desvantagens que tinham inicialmente as redes MLP era a ausência de um algoritmo de treinamento, que foi proposto com a publicação do livro de (RUMELHART; MCCLELLAND, 1986). Para o treinamento de uma MLP é utilizado um algoritmo de otimização baseado no gradiente descendente, chamado *Back-Propagation* (BP).

Segundo (HAYKIN, 1994), esse algoritmo se divide em duas fases. Uma fase para frente (forward), conhecida como "propagação", e uma fase para atrás (backward), chamada "ajuste de pesos por retro-propagação".

Para a fase de propagação, considere a Figura 12, de um diagrama de fluxo de sinal, com neurônios interconectados a um sinal de entrada x_m. Similarmente ao perceptron, cada unidade de processamento calcula a função $v_j = \sum w_{ji}x_i$, que é a saída do neurônio j.

Figura 12 – Fluxo de sinal de um neurônio.



Fonte: Adaptado de (HAYKIN, 1994)

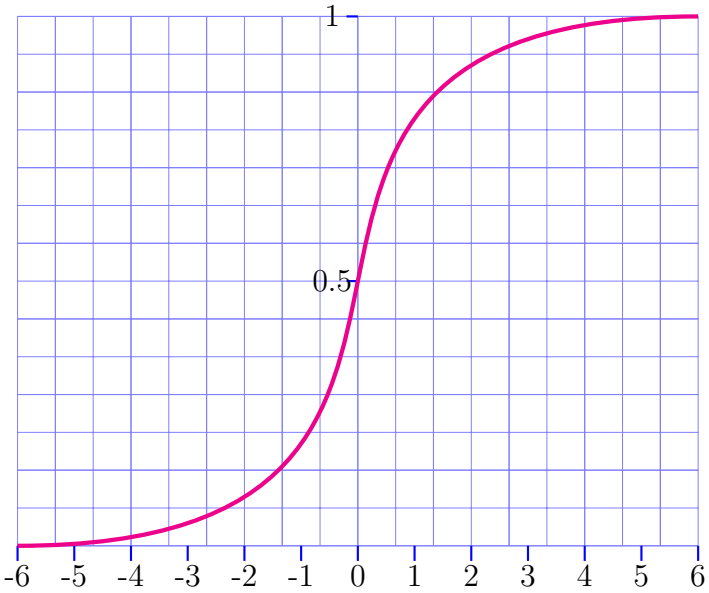
à saída v_j, aplica-se uma função de ativação $\varphi(.)$. Na equação 16, y_j é a saída do neurônio j após a aplicação da função de ativação. Cada neurônio nessa camada aplica a função ativação à sua entrada total e produz um valor de saída, que é utilizado como valor de entrada para os neurônios da camada seguinte. Esse processo continua até que os neurônios da camada de saída produzam cada um, o seu valor de saída.

$$y_j = \varphi(v_j) \tag{16}$$

A equação 17 é função de ativação mais comumente utilizada, chamada sigmóide. Perceba, conforme ilustrado na Figura 13, que essa função fornece saídas contínuas, em contraste com as saídas discretas da função limiar na Figura 8.

$$\varphi(v_j) = \frac{1}{1 + e^{-v_j}} \tag{17}$$

Figura 13 – Função sigmóide.



Fonte: Elaborada pelo autor.

Para que esse algoritmo funcione, a função de ativação precisa ser contínua, diferenciável e de preferência não decrescente. A função tipo sigmóide cumpre esses critérios, tornando possível o uso do método de Gradiente Descendente (GD) para abordar o aprendizado do MLP (MELLO; PONTI, 2018, p.52–53). .

Segundo Cerri (2013), a " Idéia básica do processo de aprendizado do *Back-Bropagation* é a repetida aplicação da regra da cadeia do Cálculo, de modo a calcular a influência de cada peso sináptico da rede no processo de aprendizagem". Isto quer dizer que a influência é calculada em relação a uma função de erro ξ com os parâmetros da rede (pesos e bias). Esse erro é uma medida de desempenho e o alvo do treinamento é minimizá-lo.

O erro é a diferença entre os valores de saída de cada neurônio na camada de saída e os valores desejados, conforme ilustrado na Figura 12.

Após obter as sinapses da camada de saída, inicia-se a fase de ajustes e pesos. Primeiro, calcula-se o erro produzido pela rede $e = d_j - y_j$. Após o cálculo, utiliza-se o Erro Quadrático Médio (18) para cada neurônio da camada de saída (j) e os exemplos da entrada x_i, como na Figura 12. Em seguida, esses valores são somados e normalizados segundo a quantidade de exemplos no conjunto de dados (19).

$$\xi(x) = \frac{1}{2}\sum_{k} e_j^2(x) \tag{18}$$

$$\xi_{medio} = \frac{1}{N}\sum_{i=0}^{N} \xi(x) \tag{19}$$

Como se mencionou anteriormente, a medida ξ é uma medida de desempenho e precisa ser minimizada. Essa minimização é feita ajustando-se os pesos ao valor da derivada

parcial $\partial\xi$ (Equação 20)

$$\frac{\partial \xi}{\partial w_{ji}} = \frac{\partial \xi}{\partial e}.\frac{\partial e}{\partial y_j}.\frac{\partial y_j}{\partial v_j}.\frac{\partial v_j}{\partial w_{ji}} \tag{20}$$

A derivada parcial representa um valor de sensibilidade e determina a direção da busca pelo peso sináptico w_{ji} no espaço de pesos (CERRI, 2013).

Após do cálculo das derivadas, o objetivo do aprendizado é a minimizar a função erro ξ por meio da regra Delta dada pela Equação 21.

$$w_{ji}(t+1) = w_{ji}(t) - \eta \frac{\partial \xi}{\partial w_{ji}} \tag{21}$$

A convergência de uma rede depende da taxa de aprendizado chamada η de regiões de convergência (CERRI, 2013) do algoritmo *Back-Bropagation*. Se a taxa têm um valor muito alto, pode provocar oscilações que evitariam a convergência. Por outro lado, se for muito baixa, requere-se muitos ciclos para que o modelo seja estável. O critério de parada pode ser número máximo de ciclos ou uma taxa máxima de erro. As regiões de convergência dependem dos mínimos locais e globais. O objetivo do treinamento do *Back Propagation* é atingir os mínimos globais sem ficar preso em um mínimo local, fazendo a rede com uma baixa acurácia preditiva (FACELI et al., 2000).

Têm casos onde o algoritmo não consegue aprender devido as configurações dos parâmetros e mostra resultados insatisfatório. Esses casos são denominados *overfitting* e *underfitting*. Uma explicação mais detalhada encontra-se mas a frente na Secção 4.5

4.3 Autoencoders

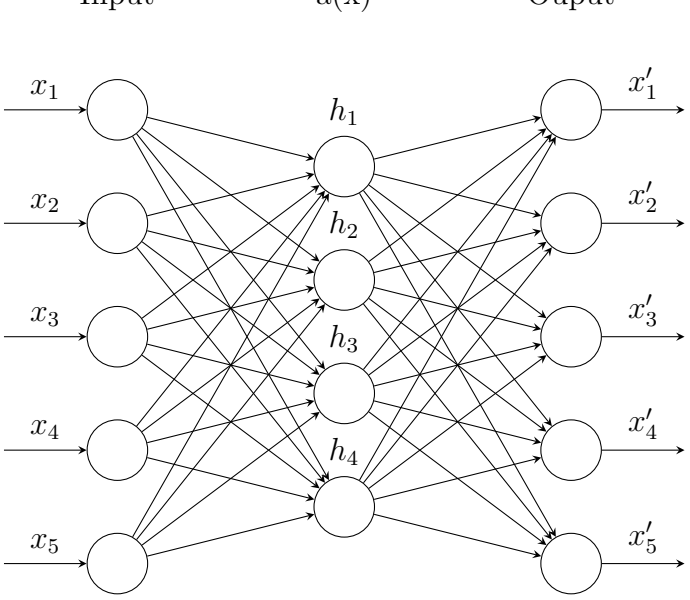
Os Autoencoders (AE) são um tipo de ANN do tipo *FeedFordward*, inicialmente propostas por Freund e Haussler (1992). A diferença entre uma MLP e um AE, é que o alvo da AE é reconstruir a entrada, enquanto a MLP busca predizer os valores de saída esperada dada entrada (LIU et al., 2017).

Os AE são uma técnica de algoritmos não supervisionados, na qual utiliza as redes neuronais para a tarefa de aprendizado de representação, especificamente força uma representação do conhecimento compactada da entrada original. Este tipo de redes utilizam o algoritmo *Back-Propagation*. Também são utilizados para a redução da dimensionalidade e compressão dos dados (LIU et al., 2017). Possuem uma camada de entrada, uma camada oculta e uma camada de saída treinadas de maneira não supervisionada.

O processo de treinamento do AE consiste em duas partes: *encode* e *decode*. O processo de *encode* mapeia a entrada de dados x para uma representação abstrata de características em uma camada oculta. O *decode* é a etapa de reconstrução a seus dados originais a partir da camada oculta. Especificamente, esse método é treinado com a função *encoding* com a entrada de dados x para transformar a algum vetor representação h_n para logo

reconstruir a partir dessa representação os dados originais x' com uma função chamada *decode*, conforme ilustrado na Figura 14.

Figura 14 – Arquitetura de um Autoencoder. O modelo recebe a entrada x e codifica em $a(x)$ para posteriormente decodificar em x'.



Fonte: Elaborada pelo autor.

O passo da codificação é definido pela Equação 22. Dado um conjunto de dados não rotulados $\{x_n\}_{n=1}^{N} \in \mathbb{R}^k$ e $x' \in \mathbb{R}^k$ é o vetor de entrada e o vetor reconstruído respectivamente. $h(x)$ é o vetor oculto gerado pela primeira camada calculado a partir de x_n. A função f é a função *encode*. Os valores W_1 e W_2 são matriz de pesos do *encoder*, b_1 é o vetor *bias*.

$$h(x) = f(W_1 x + b_1) \tag{22}$$

O processo do *decoder* é definido pela Equação 23

$$x' = g(W_2 h(x) + b_2) \tag{23}$$

O valor x' é o vetor decodificado da camada de saída. A função g é a função decodificadora, W_2 é a matriz de peso do *decoder* e b_2 é o *bias*.

A função de otimização para a minimização do erro entre o vetor de entrada e o vetor reconstruído da saída é definido pela Equação 24 (BAO; YUE; RAO, 2017).

$$\Theta = argmin \frac{1}{n} \sum_{i=1}^{n} ||x_i - x_i^{'}||^2 \tag{24}$$

Onde $||x_i - x_i^{'}||^2$ representa a função de custo (*loss function*). Os valores x_i e $x_i^{'}$ são os i-esimo valor do vetor de entrada e da reconstrução.

Esse processo permite o aprendizado de características úteis, já que o vetor de entrada é transformado em uma representação dimensional menor e há filtragem de informação inútil (LIU et al., 2017)

Nos trabalho de Vincent et al. (2008) e Vincent (2011) os autores propuseram uma variação do Autoencoders chamado *Denoising AutoEncoders* (DAE) que adiciona intencionalmente ruído nos dados de treinamento e busca transformar a entrada (ruidosa) na replicação do dado sem ruídos.

Outra arquitetura de AE foi desenvolvida em (VINCENT et al., 2010). Os *Stacked AutoEncoders* (SAE) são construídos por uma sequência de camadas empilhadas. Também seguem os mesmos passos de AE e utilizam o algoritmo BP.

4.4 Deep Learning

Um termo informal introduzido para diferenciar as vantagens e desvantagens de dois tipos de conceitos: arquiteturas "rasas" e "profundas" (BENGIO; LECUN, 2007). Durante muito tempo, foram utilizadas arquiteturas "rasas" para a modelagem de problemas. Essas arquiteturas tinham no máximo uma ou duas camadas ocultas de características não lineares. Apesar do sucesso de arquiteturas como a MLP, elas não foram suficientes para modelar diversos tipos de problemas reais, pois seu poder representativo não foi suficiente (Deng; Yu, 2014).

Devido às exigências de aplicações do mundo real, especialmente com dados não estruturados, como processamento de linguagem natural, reconhecimento de fala humana e processamento visual, surgiu a necessidade de arquiteturas profundas, com muitas camadas, em oposição a arquiteturas "rasas", para extrair estruturas complexas da representação do conhecimento. As arquiteturas profundas permitem a representação de uma ampla família de funções em forma mais compactas que as arquiteturas rasas (BENGIO; LECUN, 2007).

O aprendizado feito por essas redes profundas é conhecido como *Deep Learning* (DL). Os métodos que utilizam DL buscam descobrir uma hipótese, um modelo ou função, a partir de um conjunto de dados (exemplos) e um método para guiar o aprendizado. Ao final, o DL têm que ser capaz de fornecer uma função capaz de receber o conjunto de dados e jogar como saída uma representação adequada.

Ponti e Costa (2018) fazem uma questão interessante sobre o DL: **qual é a diferença entre ML e DL**? De acordo com os autores. "a diferença está baseada em como aprende a função $f(.)$. Os algoritmos "não-DL" ("rasos" ou "superficiais") buscam diretamente por uma função que possa, a partir de um conjunto de parâmetros, gerar um resultado. No DL, há métodos que aprendem a função $f(.)$ por meio da **composições** de funções", assim:

$$f(x) = f_L(...f_2(f_l(x_l))...))$$

A função f_l representa a primeira camada com índice l que recebe como entrada o conjunto de dados o vetor x_1, e aplicada a cada função a nova saída da função anterior a próxima função gerando o próximo vetor x_{l+1}.

Cada função faz uso de parâmetros para transformar os dados de entrada. Esse conjunto de parâmetros é normalmente uma matriz W_l relacionada a cada função f_l (PONTI; COSTA, 2018), então pode-se escrever:

$$f(x) = f_L(...f_2(f_1(x_1, W_1), W_2)...), W_L)$$

Onde x_1 representa os dados de entrada e cada função têm seu próprio conjunto de parâmetros e sua saída será passada a próxima função. Uma das idéias importantes no DL é aprender sucessivas representações dos dados ou seja os parâmetros W diretamente a partir dos dados fazendo combinações de outras funções que transformam vetores, mapeando-os de um espaço a outro até chegar ao resultado final (PONTI; COSTA, 2018).

Na continuação desta seção, apresenta-se as arquiteturas profundas utilizadas neste trabalho.

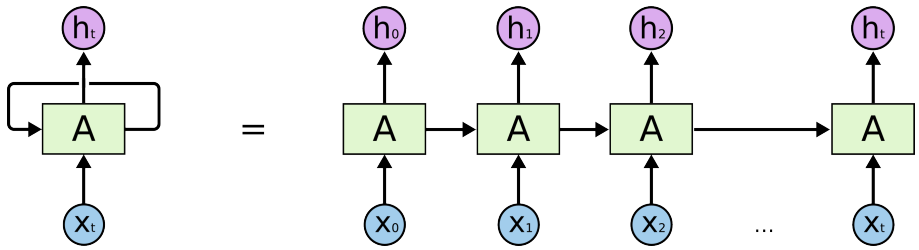
4.4.1 Redes Recorrentes

As *Recurrent Neural Network* (RNN) são uma categoria de modelos de rede neural proposta nos anos 80 (RUMELHART; MCCLELLAND, 1986; WERBOS, 1988; ELMAN, 1990). As RNN foram desenhadas para mapear sequências de dados de entrada em sequências de dados de saída. São, portanto, consideradas ideais para modelar dados com dependência sequencial, como texto, genes, fala e também para previsão de séries temporais. Essas redes foram criadas para manter informação passada para serem utilizadas no futuro.

Em uma RNN, os neurônios ocultos permitem conexões entre eles associados a um tempo de espera. Essas conexões permitem à RNN reter informação do passado para descobrir correlações "temporárias" entre eventos distantes nos dados. Esse modelo leva em conta o tempo e sequência ordenada temporalmente e destaca-se pelo tipo de memória que têm ao mesmo que o ser humano (PASCANU; MIKOLOV; BENGIO, 2013).

Quando uma pessoa está lendo um texto, ela entende a palavra atual baseando-se em palavras anteriores. Similarmente, as RNN têm *loop* internos e conexões para manter essa informação. A RNN pode ser vista como uma rede com múltiplas cópias de si mesma que passam mensagem ao seu sucessor ao longo do tempo. Na figura 15, mostra-se uma rede neural (A) no lado esquerdo, com uma entrada x e a cada passo x_t, para cada entrada temos a saída h_t. No lado direito, está o mesmo processo da rede desenrolada. O *loop* permite que a informação seja persistente passando informação a si mesma.

Figura 15 – Modelo de Rede Neural Recorrente.



Fonte: Olah (2018).

Uma das vantagens de uma RNN é o fato de que elas são capazes de utilizar informação prévia para a tarefa atual. No entanto, enquanto existem alguns problemas que precisam verificar a informação recente para tomar decisões, existem outros que precisam conhecer o contexto passado. Uma desvantagem das RNN é que para aprender essas informações passadas elas precisam uma lacuna muito grande entre o presente e o passado e, várias vezes, não são capazes de conectar essa informação. Na seção a seguir será melhor explicado tal problema.

4.4.1.1 O problema das Dependências *Long-Term*

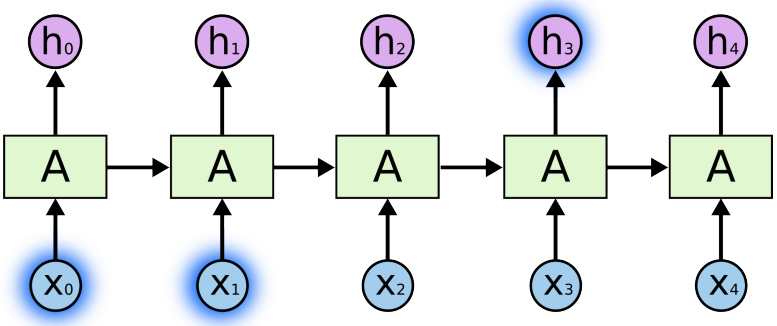
Uma das características das RNN é que elas podem conectar informações anteriores a uma tarefa atual. Mas realmente podem? Por exemplo, considere em um modelo de linguagem em que precisamos prever a última palavra baseando-se nas palavras anteriores "As nuvens estão no céu". Nesse caso não precisamos do contexto passado e a última palavra seria "céu". Existem outros casos onde precisamos saber mais do contexto e a lacuna entre a informação relevante e onde ela é requerida é pequena. Nesse caso, as RNN ainda podem aprender o conceito necessário, mas existem casos em que precisamos saber mais sobre o contexto e a lacuna é muito grande.

Considere que agora tenta-se prever a última palavra na frase "eu sou do Brasil .... eu falo português". Na Figura 16(b), observa-se que a lacuna é maior entre a informação relevante e onde ela é precisada enquanto na Figura 16(a) está muito perto. As desvantagens das RNN acontece nesses casos, em que não conseguem, na prática, conectar essas informações de dependência longa.

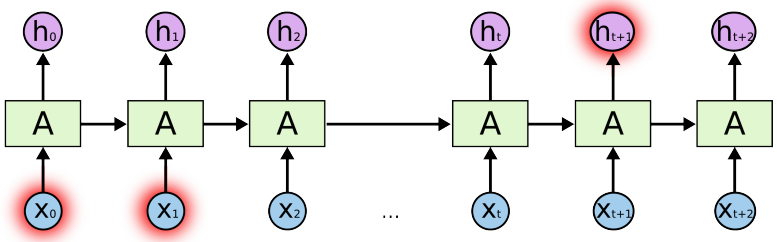
Em teoria, as RNN são capazes de aprender estas dependências "*long-term*". Porém, não conseguem aprender, na prática. Todos esses problemas foram explorados por Hochreiter (1991) e Bengio, Simard e Frasconi (1994). Os primeiros autores mostraram que as redes *Feedforward Neural Network* (FFNN) e RNN são difíceis de treinar com o algoritmo *Back Propagation*. Seu trabalho identificou que as arquiteturas profundas sofrem do problema do *vanishing gradient* e *exploding gradient*. O modelo da RNN é muito simples e poderoso, mas muito sensível aos problemas citados.

O problema que existia na academia era o *Back-Propagation Through Time* (BPTT)

Figura 16 – Dependências de informações na RNN.



(a) A informação relevante está perto de h_3.



(b) A informação relevante têm uma lacuna muito longa.

Fonte: Olah (2018).

ou *Real-Time Recurrent Learning* (RTLR). Sinais de erro que "fluem para trás no tempo" tendem a 1.) explodir (exploding) ou 2.) desaparecer (vanish). A evolução temporal do erro exponencialmente retro-propagado depende do tamanho dos pesos da matriz (HOCHREITER; SCHMIDHUBER, 1997).

Para Bengio, Simard e Frasconi (1994), esses problemas acontecem quando:

a) **Exploding gradient**, ocorre quando existe um incremento gigante na norma do gradiente na fase do treino. Tais eventos devem-se à explosão dos componentes de *long-term*, que podem crescer exponencialmente mais do que os de *short-term*.

b) **Vanish gradient**, é oposto ao problema anterior, ou seja, os componentes *long-term* crescem exponencialmente para a norma 0 tornando impossível ao modelo aprender as correlações entre eventos temporalmente distantes.

4.4.2 Long Short-Term Memory

Em meados dos anos 1990, uma variação das RNN, chamadas redes *Long Short-Term Memory* (LSTM), foi introduzida por Hochreiter e Schmidhuber (1997). As LSTM são capazes de aprender dependências "*long-term*" e, por isso, posteriormente foram popularizadas em vários trabalhos.

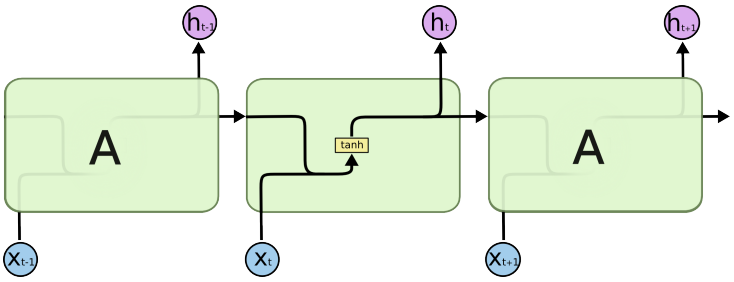
Para superar os problemas de *vanishing* e *exploding*, Hochreiter e Schmidhuber (1997) propôs uma nova arquitetura, que força a manter um erro constante mediante estados internos e unidades especiais chamadas *gates*. A diferença básica da LSTM com outras

redes é que algumas unidades chamadas *Constant Error Carousel* (CEC), utilizam uma função de ativação f que está conectada a si mesma com pesos iguais a 1.0. Devido a isso e ao fato de que a derivada de uma constante é 1.0, o erro não pode se propagar através de CEC. Consequentemente, não há *vanish* e *exploding* (SCHMIDHUBER, 2015). As CEC são umas das razões porque uma LSTM consegue aprender a memorizar eventos que aconteceram no passado, enquanto as RNN falham com 10 passos (SCHMIDHUBER, 2015).

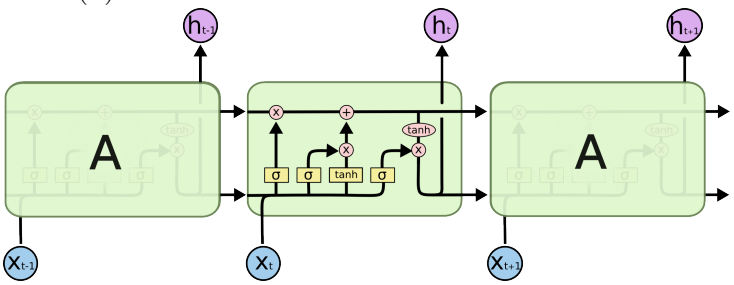
A LSTM têm uma arquitetura parecida com as RNN. Elas foram desenhadas especificamente para evitar o problema da dependência conhecida como "*long-term*" e reter informação ao longo do tempo. Todas as redes neurais recorrentes têm uma estrutura de forma repetitiva, a primeira camada é uma camada com uma função *tanh*.

A principal diferença com a LSTM, é que os módulos são repetitivos, mas no lugar de ter uma camada, as LSTM têm quatro camadas interiores com *sigmóide*, *sigmóide*, *tanh*, *sigmóide*. Na Figura 17(a), representa uma RNN e na Figura 17(b), mostra-se a LSTM com quatro diferentes camadas.

Figura 17 – Diferentes estruturas internas da RNN e a LSTM.



(a) A RNN com uma camada no interior.



(b) A LSTM contém quatro camadas interagindo

Fonte: Olah (2018)

A LSTM é composta de uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. A característica mais importante de uma LSTM, que a diferencia das RNN, está na camada oculta. Especificamente, nas células de memória. Cada célula de memória contém três estruturas *gates* e uma célula estado. Os estados das célula são regulados pelos *gates* (HOCHREITER; SCHMIDHUBER, 1997):

- **Forget gate**, que define quais informações são removidas e quais serão mantidas no estado da célula.

- **Input gate**, que especifica quais informações são adicionadas ao estado da célula.
- **Output gate**, que especifica quais informações do estado da célula são usadas como saída.

Os *gates* são mecanismos responsáveis por regular a passagem das informações. Eles são compostos de uma camada "sigmóide" e uma operação de multiplicação ponto. A função sigmóide é um valor entre $0 - 1$. O valor de zero significa "que não deixe passar nada" e enquanto o valor de 1 significa "deixe tudo passar!".

Antes de explicar os passos, explicaremos a notação utilizada.

- O valor x_t é um vetor de entrada no passo t
- Os W_f, W_i, W_C são matrizes de pesos
- Os valores b_f, b_i, b_C são vetores bias.
- Os f_t, i_t, o_t são os valores de ativação para cada *gate*.
- C_t e $\widehat{C}_t$ são vetores para valores do estado da célula.
- h_t e h_{t-1} é o vetor da saída da LSTM.

O primeiro passo é decidir que informações vão descartar o estado da célula. Para isso, utiliza-se a função sigmóide. Na Equação 25, é apresentado o *forget gate*, que recebe como entrada o valor x_t, a saída anterior h_{t-1}, os pesos W_f e o bias b_f. A saída dessa função é o valor 1 para decidir se o valor do estado anterior C_{t-1} é retido ou 0 se vai se livrar (esquecer) completamente disso.

$$f_t = sigmoide(W_f.[h_{t-1}, x_t] + b_f) \tag{25}$$

O seguinte passo é decidir qual informação vai se atualizar no estado da célula C_t, o que é realizado em duas etapas. A Equação 26 mostra o *input gate*, sendo que a função sigmóide é responsável por decidir que valores serão atualizados. Depois, a Equação 27 define a camada da *tanh*, que cria um novo candidato $\widehat{C}_t$, que será adicionado ao estado. Finalmente, serão combinados com o valor i_t. O novo estado C_t é definido pela Equação 28, que é a soma do estado antigo C_{t-1} por f_t ao novo candidato do estado da célula $i_t * \widehat{C}_t$. As Figuras 18(b) e 18(c) ilustram esse processo.

$$i_t = sigmoide(W_i.[h_{t-1}, x_t] + b_i) \tag{26}$$

$$\widehat{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C) \tag{27}$$

$$C_t = f_t * C_{t-1} + i_t * \widehat{C}_t \tag{28}$$

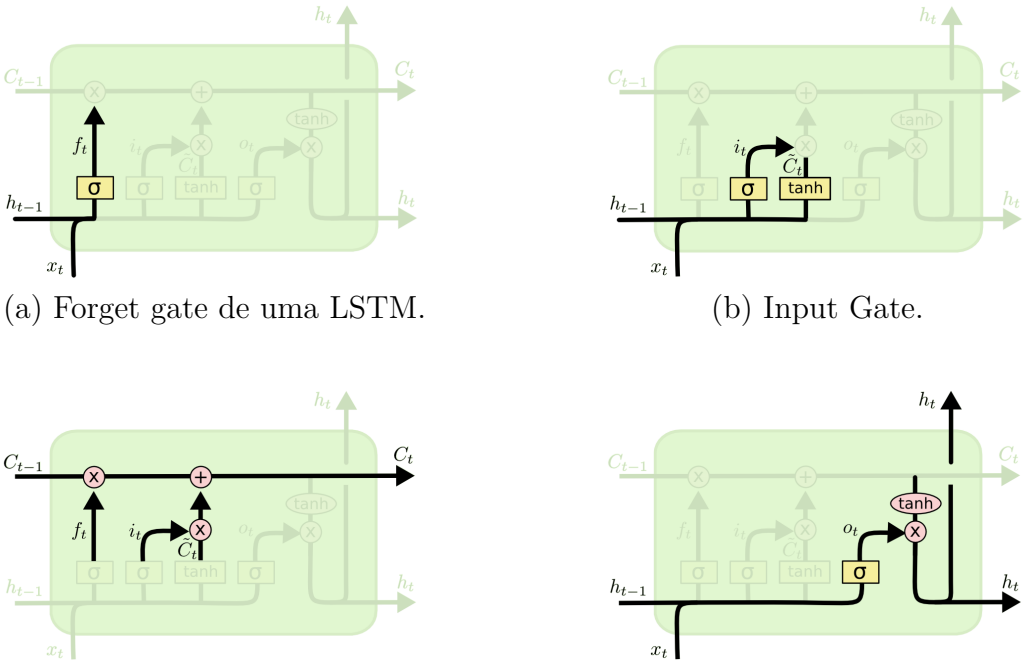
Finalmente, precisamos um valor de saída da rede. Esse processo é feito pelo *gate* de saída. A Equação 29 define o *gate* de saída, formada pela multiplicação do valor da função sigmóide e o valor do estado da célula C_t, aplicado um filtro $tanh(C_t)$ entre $[-1, 1]$. O filtro é definido pela Equação 30.

$$o_t = sigmoide(W_o.[h_{t-1}, x_t] + b_o) \tag{29}$$

$$h_t = o_t * tanh(C_t) \tag{30}$$

A Figura 18(d) ilustra o funcionamento do *output gate* da rede LSTM.

Figura 18 – Funcionamento interno de uma LSTM.



(a) Forget gate de uma LSTM. (b) Input Gate.

(c) Célula de estado. (d) Output Gate.

Fonte: Olah (2018)

4.4.3 Mecanismos de Atenção

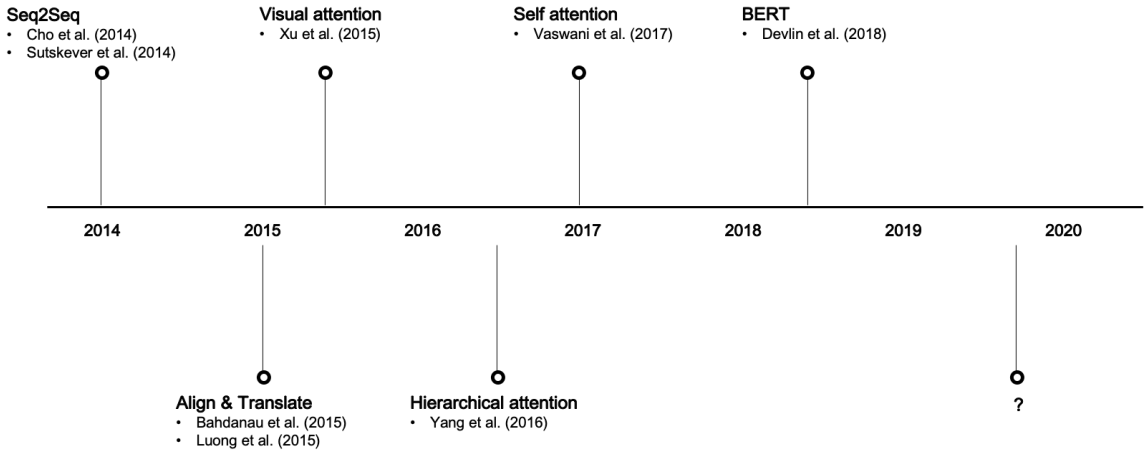
O modelo de Atenção é um dos conceitos mais poderosos no campo do aprendizado profundo hoje em dia. Ele é baseado numa intuição de senso comum onde "prestamos atenção a" uma determinada parte ao processar uma grande quantidade de informação. Este simples conceito revolucionou não só o Processamento de Língua Natural (PLN) também outras áreas como processamento de imagens, reconhecimento de fala e sistemas de recomendação.

Este paradigma se tornou muito popular na comunidade de Inteligencia Artificial como uma arquitetura essencial em PLN. Foi introduzido pela primeira vez em Bahdanau, Cho e Bengio (2015) e é baseado em sistemas biológicos humanos. Por exemplo, nosso sistema

de processamento visual tende a se concentrar seletivamente em algumas partes de uma imagem, enquanto ignora outras informações irrelevantes de uma maneira que pode ajudar na percepção (XU et al., 2015). Similarmente, pode-se aplicar em outro tipo de tarefas que envolvem processamento de linguagem, reconhecimento de fala e voz, somente algumas partes da entrada são mais importantes que outras, ou seja nosso cérebro se enfoca mais em partes relevantes.

O avanço desta arquitetura foi marcado por trabalhos importantes que contribuíram seu progresso, a Figura 19 mostra os principais trabalhos publicados sobre o assunto.

Figura 19 – Trabalhos que contribuíram ao avanço da pesquisa com modelos baseados em atenção.



Fonte: Kim (2020)

No início, o modelo *seq2seq* (sequência para sequência) proposto por Cho et al. (2014) e Sutskever, Vinyals e Le (2014), é um tipo de arquitetura para resolver problemas de tradução de máquina. A *seq2seq* é baseada nas redes recorrentes RNN. Normalmente são formados por uma arquitetura *encoder-decoder* (originalmente o mecanismo de Atenção utilizavam esta arquitetura) (SHIH; SUN; LEE, 2018). Elas foram usadas para resolver problemas complexos como: tradução automática, chat-bots e resumo do texto. Na prática, utilizam as variantes de RNN do tipo LSTM (HOCHREITER; SCHMIDHUBER, 1997) e *Gated Recurrent Unit* (GRU) Neural Networks (CHUNG et al., 2014).

A maioria dos modelos para traduzir idiomas pertenciam a famílias de *encoder-decoder*, um *encoder* e *decoder* para cada linguagem. O *encoder* processa a sequência de entrada para comprimi-la em um vetor de contexto com tamanho fixo (BAHDANAU; CHO; BENGIO, 2015). O *decoder* pega essa saída do vetor de contexto para decodificá-lo e utiliza para gerar a sequencia de saída. Esses modelos obtiveram resultados muito relevantes para a área, porém, eles contém limitações que prejudicam seu desempenho.

Um problema comum nas arquiteturas *seq2seq* é que não conseguiam capturar algumas informações quando usam um vetor final de tamanho fixo (h_t). Isso pode ser especialmente problemático quando se está processando frases longas em que a RNN é incapaz de

enviar informações adequadas ao final das frases devido ao *gradient exploding*. Portanto, Bahdanau, Cho e Bengio (2015) propuseram utilizar um vetor de contexto para alinhar as entradas de origem e destino. Este vetor de contexto preserva informações de todos os estados ocultos das células do *encoder* e os alinha com a saída de destino atual. Ao fazer isso, o modelo é capaz de "atender" a uma determinada parte das entradas de origem e aprender melhor a relação complexa entre a origem e o destino.

A arquitetura da rede neural *Transformer*, proposta por Vaswani et al. (2017), marcou um dos maiores avanços da década no campo do PLN. Este modelo é baseado em mecanismos de atenção, que contém uma camada de *self-attention* com *multi-head*. O Transformer alinha as palavras com outras na sequência, calculando assim uma representação dessa sequência. A Figura 20, mostra internamente como é composta essa arquitetura. Não é apenas mais eficaz na representação, mas também mais eficiente computacionalmente em comparação com operações de convolução e recursivas, permitindo a paralelização.

Esta arquitetura inspirou outras pesquisas, liderando o desenvolvimento de novos modelos baseados em *self-attention*, como o *Bidirectional Encoder Representations from Transformers* (BERT) (DEVLIN et al., 2019), que hoje é o estado da arte para várias tarefas de PLN. O modelo BERT será explicado na Seção 5.6.3.

4.5 Conceitos Adicionais

Nesta seção, serão apresentados conceitos importantes para a avaliação de um algoritmo de AM supervisionado utilizados neste trabalho. As métricas utilizadas serão: as métricas de classificação e regressão que medem o desempenho de um preditor, além disso conceitos importantes para evitar que os modelos estejam muito ajustados aos dados como é o *overfitting* e *underfitting*.

Overfitting Esse fenômeno ocorre quando um classificador busca modelar excessivamente o conjunto de treino na entrada de dados, resultando em taxas de erro baixas no conjunto de treino, porém altas no conjunto de teste (MURPHY, 2012).

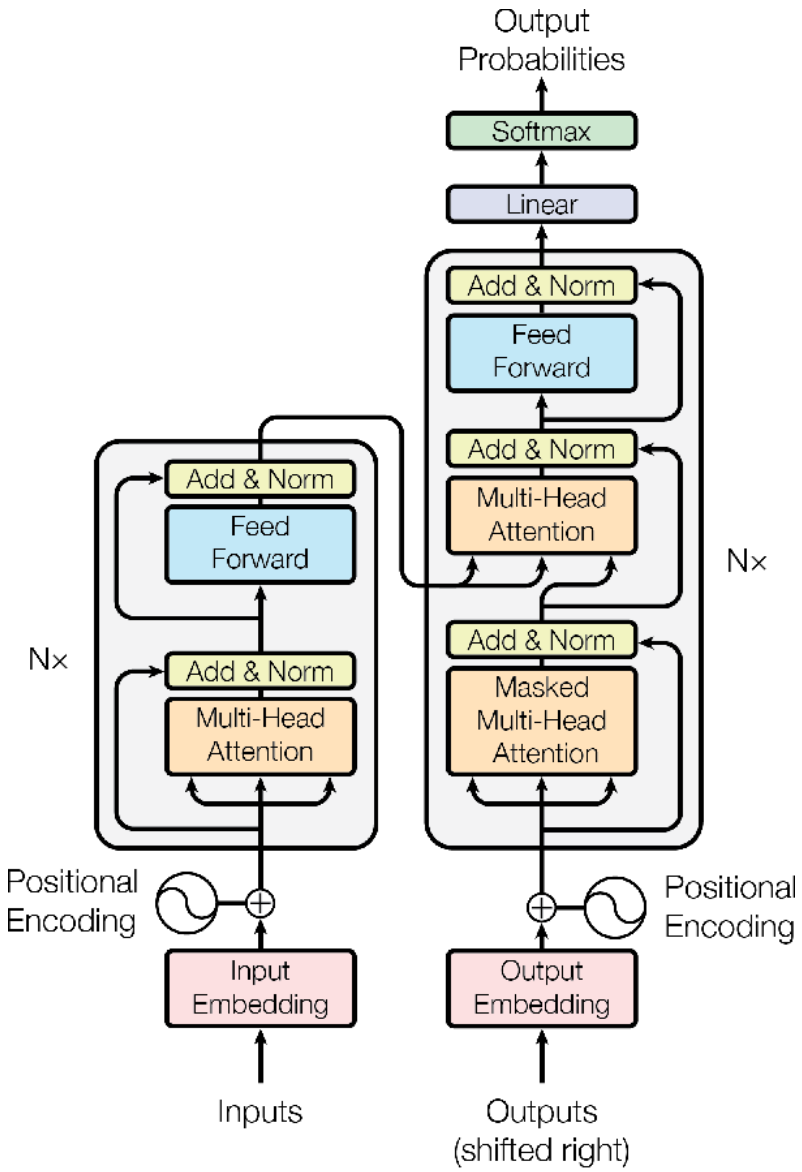
Underfitting O problema de *underfitting* ocorre quando um classificador não consegue aprender a distribuição dos dados, resultando em taxas de erro altas para ambos conjuntos de teste e treino.

4.5.1 Medidas de avaliação

Para a compreensão dos resultados obtidos, os modelos de previsão de séries temporais precisam ser avaliados. Existem medidas estabelecidas para medir a capacidade preditiva, que podem ser úteis para problemas de classificação ou para tarefas de regressão.

Para Faceli et al. (2000), a avaliação de um algoritmo de ML supervisionado é normalmente feita por uma análise do desempenho do preditor para se rotular novos objetos, não

Figura 20 – Arquitetura do Transformer.



Fonte: Vaswani et al. (2017)

apresentados na sua base de treinamento. Nesta pesquisa, serão utilizadas as seguintes medidas:

1. **Para Regressão**: O erro da função hipótese $\hat{f}$ pode ser calculado pela distância entre o valor $\mathbf{y}_i$ conhecido e o valor predito por $\hat{\mathbf{f}}(x_i)$. As medidas mais comuns para esse fim são, segundo (FACELI et al., 2000):

   MSE Mean Squared Error, mede a distância entre os valores verdadeiros e preditos, com maior destaque para erros maiores, proporcionado pela segunda potência.

$$\text{MSE} = \frac{\sum_{i=1}^{N}(\mathbf{y}_i - \hat{\mathbf{f}}(x_i))^2}{N} \tag{31}$$

MAPE Mean Average Percent Error, apresenta erros de previsão em percentagem.

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\mathbf{y}_i - \hat{\mathbf{f}}(x_i)\right|}{|\mathbf{f}(x_i)|} \times 100 \tag{32}$$

MAE Mean Average Error, mede a diferença absoluta média entre as previsões e o valor real.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|\mathbf{y}_i - \hat{\mathbf{f}}(x_i)\right| \tag{33}$$

RMSE Root Mean Square Error, também mede a distância entre as previsões e os valores reais, mas, ao contrário do MAE, grandes desvios do valor real têm um grande impacto no RMSE devido ao erro de previsão quadrática.

$$\text{RMSE} = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}_i - \hat{\mathbf{f}}(x_i))^2} \tag{34}$$

R-square R^2, avalia a dispersão dos pontos de dados ao redor da linha de regressão. É também chamado de coeficiente de determinação. O R^2 é a porcentagem da variação da variável, é usado para analisar como as diferenças em uma variável podem ser explicadas por uma diferença em uma segunda variável.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{35}$$

Onde:

$$SS_{res} = \sum_{i=1}^{N}(\mathbf{y}_i - \hat{\mathbf{f}}(x_i))^2$$

$$SS_{tot} = \sum_{i=1}^{N}(\mathbf{y}_i - \hat{x})^2$$

O valor $\hat{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ é a média. O R^2 sempre é um valor entre $0 - 1$.

Capítulo 5

Processamento de Língua Natural

O Processamento de Língua Natural (PLN) é uma área da computação que tem como objetivo extrair representações de textos livres escritos em língua natural como objeto primário de estudos. Neste capítulo, aborda-se como o PLN busca, em geral, encontrar soluções para problemas computacionais que requerem o tratamento computacional de alguma língua natural que seja escrita ou falada.

5.1 Análise de Sentimentos

A Análise de Sentimentos (AS), também chamada de mineração de opinião, é um campo ativo de estudo dentro *Processamento de Línguas Naturais* (PLN) que analisa opiniões, sentimentos, avaliações, atitudes e emoções das pessoas em relação a produtos, serviços, organizações, indivíduos, questões, eventos, tópicos e seus atributos. A análise de opinião e a mineração de opinião enfocam-se, principalmente, nas opiniões que expressam sentimentos positivos ou negativos (LIU, 2012).

5.1.1 Diferentes Níveis de Análise

Um dos principais problemas que o PLN têm que lidar são os diferentes níveis de análise que têm que realizar. Em geral, existem três níveis baseados na granularidade abordada: documento, sentença e aspecto (LIU, 2012).

1. **Nível de Documento.** A tarefa neste nível considera o texto como um todo e classifica toda a opinião do documento como "positiva" o "negativa". Esta tarefa é conhecida como *document level classification*, que faz a análise do sentimento em

relação a uma "entidade" e não aplica a documentos que fazem uma comparação entre várias entidades.

2. **Nível de Sentença.** Este tipo de tarefa realiza uma análise a nível de sentença e determina se é positiva, negativa ou neutra.

3. **Nível de Entidade ou Aspectos.** A análise no nível documento e sentença não conseguem dizer o que as pessoas gostam e que não gostam. A análise em nível de entidade analisa diretamente a opinião por si mesma, que está baseada no sentimento (positivo, negativo) e um objetivo (de opinião). Em várias aplicações, o alvo de opinião são descritas por entidades ou seus diferentes aspectos. O alvo desta análise é descobrir sentimentos em entidades. Por exemplo, considere a sentença *"the iphone's calls quality is good, but its battery life is short"*. Nela, observa-se que quem a escreveu avalia dois aspectos do aparelho: a qualidade da ligação e o tempo de vida da bateria. Esses constituem os aspectos, o alvo das opiniões.

A definição de como usar cada um desses níveis depende do origem dos dados que utiliza-se (SANTOS, 2019). Por exemplo, se utilizamos uma fonte de dados de um jornal, eles são completamente diferente de uma fonte de dados de *Twitter*. Os jornais são texto mais cumpridos e com uma língua culta. No caso de *Twitter*, utiliza-se uma língua informal e muito curta. Portanto, normalmente, utiliza-se o nível de sentença.

5.2 Linguística do Corpus

O AS precisa de recursos para realizar as diferentes tarefas associadas a ele. Um dos principais recursos é o corpus, como uma forma de representação dos dados naturais do discurso da pessoa para a construção do PLN.

Existem várias definições para Corpus. Para Sardinha (2000): são textos naturais com as características de naturalidade e autenticidade. Isto quer dizer, aqueles que existem na linguagem estudada e que não foram criados por um processo de um computador. A idéia de natural significa aqueles que só foram criados por humanos.

O Corpus pode ser usado em diferentes aplicações, sendo que é um artefato produzido para uma pesquisa. Para Sardinha (2000 apud PERCY; MEYER; LANCASHIRE, 1996, p. 15-16), "corpus é uma coletânea de porções de linguagem que são selecionadas e organizadas de acordo com critérios linguísticos explícitos, a fim de serem usadas como uma amostra da linguagem". Isso remete aos problemas relacionados à delimitação do texto como um trecho de conversação, um resumo de um artigo etc.

Uma definição mais exata para corpus menciona o seguinte:

> "Um conjunto de dados linguísticos (pertencentes ao uso oral ou escrito da língua, ou a ambos), sistematizados segundo determinados critérios, suficientemente extensos em

> amplitude e profundidade, de maneira que sejam representativos da totalidade do uso linguístico ou de algum de seus âmbitos, dispostos de tal modo que possam ser processados por computador, com a finalidade de propiciar resultados vários e úteis para a descrição e análise." (SÁNCHEZ, 1995, p. 8-9)

5.2.1 Construção do Corpus

A anotação de um corpus, normalmente chamada de *tagging*, pode ser definida como o processo de enriquecer o texto adicionando-se informações linguísticas e outras, inseridas por humanos capacitados na tarefa ou por computadores.

Nenhuma das duas é infalível e cada uma têm vantagens e desvantagens. No caso de anotação por humanos, ela pode ser feita num corpus pequeno. Porém, a anotação manual pode ser lenta e conter erros. Enquanto a anotação automática requer considerável quantidade de tempo na preparação e programação de um sistema de *tagging* automático de corpus, em especialmente se for treinado para depois ser usado em um corpus maior. Esses tipos são rápidos e pode produzir resultados sobe corpus muito maiores em muito pouco tempo (HOVY; LAVID, 2010).

A anotação automatizada, por outro lado, requer um investimento considerável na preparação de corpus e na programação do sistema automatizado de marcação, especialmente se for primeiramente treinado em um corpus de sementes e aplicado a um maior. Seus resultados podem ser de baixa qualidade. Mas é rápido e pode produzir resultados sobre o corpus muito grande de todos os tipos em muito pouco tempo.

A linguística do Corpus trabalha dentro de um quadro conceitual formado por abordagens "empiristas" e uma visão probabilística da linguagem . O empirismo, de jeito simples, é uma corrente filosófica segundo o qual o conhecimento se origina da experiência. Na linguística, o empirismo quer dizer que o conhecimento vem da observação da linguagem. Por outro lado, e em contraposição, existe o racionalismo, que menciona que o conhecimento provém de princípios estabelecidos a *priori*. Na linguística, o racionalismo se fundamenta no estudo da linguagem mediante a introspecção linguística como um modo de processamento do cognitivo da linguagem (SARDINHA, 2000).

Para a construção do corpus, existe uma tendência empirista que é o ponto de encontro entre PLN e a linguística do corpus. Esse empirismo é realizado por humanos e sua construção obedece a 7 questões de pesquisa de acordo com Hovy e Lavid (2010):

1. **Seleção do corpus** refere-se à preparação da amostra de textos como material inicial. Esse texto deve ter um gênero textual, pertencer a um domínio e estar balanceado.

2. **Instanciação da teoria** refere-se a especificar um conjunto de categorias, ou etiquetas, que serão utilizadas. Por exemplo as amostras dos textos, na tarefa de AS, terão as classes: positivo, neutro e negativo.

3. **Seleção e treinamento de anotadores** deve responder a questões como: a experiência dos anotadores em um tema específico, a quantidade de anotadores, similaridade de educação entre eles, o treinamento deles para anotar e se existe algum tipo de recompensa por esta tarefa.

4. **Especificação do procedimento de anotação**, fase que refere-se a anotações preliminares, ou seja, um conjunto de exemplos para ensinar aos anotadores se há algum tipo de discordância entre eles e como trataram a discordância com um juiz. Também, se existirá algum cronograma para realizar a anotação.

5. **Projeto da interface de anotação** relaciona-se à facilidade para anotar, utilizando algum tipo de *software* para ajudar na velocidade e evitar determinados tipos de viés.

6. **Escolha e aplicação de medidas de avaliação**, questão das mais importantes, devido a que se procura avaliar o grau de concordância entre dois ou mais anotadores. Uma vez terminada a tarefa da anotação, é necessário validar a confiabilidade antes de submeter a algum tipo de processamento ou algoritmo. Normalmente, existem diversas técnicas e a escolha depende dos dados. Entre elas, estão o coeficiente *Kappa* e o coeficiente de correlação de *Pearson*, utilizados para garantir a confiabilidade do corpus (DOSCIATTI; FERREIRA; PARAISO, 2015).

7. **Disponibilização e manutenção do produto** refere-se à publicação do corpus, se ele será público ou privado.

5.3 Pré-processamento do Texto

Após a coleta do texto, este deve passar por algum tipo de limpeza e filtragem, para a posterior aplicação das técnicas de aprendizado. A preparação do corpus é crucial para garantir o correto aprendizado. Segundo Benabdallah, Abderrahim e Abderrahim (2017), nesta fase há um conjunto de passos de pré-processamento para a remoção de qualquer tipo de ambiguidade e reduzir a quantidade de processamento futuro e adaptar o corpus ao objetivo final que é a "extração de termos candidatos". Esta preparação está dividida em:

1. **Normalização** é uma técnica que converte o documento em um formato padrão da língua culta, mais facilmente manipulável. Comumente, os textos não seguem as normas de uma língua (no caso do *Twitter*, por exemplo), sendo necessário remoção de caracteres especiais (sinais de pontuação, hífens e números), entre outros.

2. **Tokenização** também conhecida como segmentação de palavras, esse processo consiste em quebrar a sequência de caracteres em um texto, localizando o limite de cada

palavra, ou seja, os pontos onde uma palavra termina e a outra começa. Nas línguas europeias geralmente estão delimitadas por espaços em branco. Existem, entretanto, linguagens não segmentados por espaços, como o chinês e o tailandês, onde as palavras são escritas sem nenhuma indicação de limite entre palavras (LEANDRO et al., 2017). Considerando o exemplo abaixo, esses delimitadores normalmente são espaços entre palavras, sinais de pontuação:

"A melhor comida da América do Sul é do Peru!"

[A] [melhor] [comida] [da] [América] [do] [Sul] [é] [do] [Peru]

Nesse processo, o espaço foi considerado para separar as palavras. Ainda, se percebe especial cuidado com a remoção do caráter "!".

3. **Remoção de *stopwords***. *Stopwords* são palavras que podem ser consideradas irrelevantes para o entendimento de um texto. Esse processo elimina todo o ruído nos textos que não agregam valor semântico de maneira isolada, só de maneira global. Nessa fase, realiza-se uma comparação de palavras com as *stopwords*. As *stopwords* estão definidas em uma lista chamada "stoplist" e são: conjunções, pronomes, artigos. Exemplos: as, e, os, para, de, com, sem. Nesse passo, é comum se aplicar as curvas de **Zipf**, que medem a distribuição das frequências das palavras para ver graficamente quais acontecem com maior ou menor frequência no corpus.

4. **Stemming e lemmatization** o objetivo deste processo é reduzir a palavra a sua forma canônica do *lemma* radical mais básico ou comum. O *Stemming* geralmente corta as extremidades das palavras e geralmente inclui a remoção de afixos das palavras. A lematização geralmente refere-se ao uso apropriado de um vocabulário e análise morfológico de palavras, normalmente remove apenas as terminações flexionais e retorna a base canônica (forma em que as palavras se encontram no dicionário) de uma palavra conhecida como *lemma*.

   Por exemplo se aplica-se sobre o verbo "Entregar", "Entregando", "Entregaria" o *lemma* seria "Entreg" pois a partir do *stem* podem ser criadas outras palavras. No caso do *stemming* a palavra "saw" seria reduzida a "s" mas a lematização tentaria retornar "saw" ou "see" dependendo se a palavra for um verbo ou substantivo.

5.4 Representação do Texto

Depois da preparação do corpus, os dados obtidos são dados textuais ainda representados em língua natural. Para que o computador consiga entender um texto, ele precisa

de uma representação dos textos, em um vetor de características próprias que um algoritmo consiga entender. Antes de aprofundar é preciso compreender que a unidade fundamental de PLN é a palavra que é utilizada em representações de texto e em algoritmos. Em Linguagem natural, podemos considerar a palavra como unidade básica de significado. Por enquanto, é muito importante estabelecer representações matemáticas adequadas para que os dados textuais sejam possível processá-los e representá-los por algoritmos de Aprendizagem de Máquina automática(GOMES; EVSUKOFF, 2019).

Existem várias técnicas muito comuns para a extração de características e redução da dimensionalidade usadas na academia para a representações de texto. Algumas já pouco usadas e outras causaram um salto no estado da arte, entre elas temos: *one-hot encoding, tf-idf* e vetores densos ou *embeddings*.

No passado, abordagens tradicionais utilizavam a técnica *one-hot encoding* por ser definida como a representação de palavras categóricas em vetores esparsos ou um vetor binário, de forma que cada palavra é representada uma dimensão desse vetor (GOLDBERG, 2016) ele não contém algum tipo de afirmação de frequência. Nas próximas seções explicaremos mais detalhadamente as técnicas *tf-idf* e vetores densos ou *embeddings*.

5.4.1 Bag of Words

A técnica *Bag of Words* (BoW) é um modelo de representação do texto simples baseado num critério de frequência. Esta técnica, cada documento, texto ou sentença é pensada como um saco de palavras onde as palavras são listadas numa matriz. Cada coluna é um atributo dimensionais únicos associados à cada termo do seu vocabulário, normalmente a representação matricial gerada pela BoW é um documento-termo, onde cada documento é uma linha, cada coluna corresponde ao termo, as células indicam a relevância do termo no documento como binário(ausência, pertencer), frequência.

O BoW apresenta problemas como: a alta dimensionalidade e falta de representação semântica. Técnicas de extração de padrões são prejudicadas na presença da alta dimensionalidade. A pobreza semântica do BoW acontece devido a que a ordem das palavras não importa (CORRÊA et al., 2012).

5.4.2 Term Frequency Inverse Document Frequency

A técnica *Term Frequency Inverse Document Frequency* (TF–IDF) é um método estatístico que busca dizer quão importante é uma palavra no documento dentro de um corpus. A Equação 36 definem essas estatísticas (FELDMAN; SANGER, 2006).

$$\text{TF-IDF}(w, d) = \text{TermFrequency(w, d)} * \log(\frac{N}{DocFreq(w)}) \tag{36}$$

O TermFrequency(w, d) é o número de vezes do termo w que aparece no documento d, N é o número total de todos os documentos e o DocFreq(w) é o número total de documentos contendo o termo w.

5.4.3 Representações Distribuídas: Word Embeddings

Os vocabulários de um corpus normalmente contém milhões de palavras o suficiente para ocasionar o problema da explosão de dimensionalidade, tornando ineficiente o processamento de vetores muito grandes. Para que os algoritmos de PLN consigam uma boa capacidade de generalização, é desejável que o conhecer noções de similaridade entre palavras ainda mais se mudam de contexto (GOMES; EVSUKOFF, 2019)

Um salto importante no avanço da literatura foram a substituição de modelos lineares com entradas esparsas(one-hot) a modelos de rede neurais com representações com vetores densos contínuos com dimensões predefinidas(embeddings). Ou seja, cada característica é embutida em um espaço d-dimensional e representada como um vetor naquele espaço (GOLDBERG, 2016).

As representações distribuídas do termo em inglês "distributed word representation" são palavras atribuídas através de um contexto, induzidas a partir de dados textuais de exemplo, treinadas com métodos não supervisionados. Esses algoritmos de vetorização de palavras processam corpus gigantes para obter representações de vetores densos de valores reais, capazes de capturar as relações básicas de uma língua como a sintaxe e a semântica (GOLDBERG, 2016).

Pode-se dizer que os algoritmos de *word embeddings* são métodos de representação de palavras que reduz a dimensionalidade baseados na hipótese distribucional que consiste em considerar que palavras com significados semelhantes tendem a aparecer em contextos similares (HARRIS, 1954; SAHLGREN, 2008).

Segundo Gomes e Evsukoff (2019) "Dessa forma, termos similares tendem a ter seus vetores posicionados em uma mesma região de vizinhança no espaço vetorial criado. Portanto, essas relações de similaridade podem ser inferidas a partir do cálculo da distância entre seus vetores e a distância cosseno atua como uma métrica de similaridade entre dois termos considerando o ângulo entre seus vetores". Para similaridades entre vetores veja na secção 5.5.2.

Diferentes algoritmos foram propostos para gerar *embeddings* (BENGIO et al., 2003), (MIKOLOV et al., 2013a), (LAI et al., 2015). Essas representações podem se dividir em duas famílias. A primeira é composta de métodos que trabalham com a co-ocorrência de matriz de palavra como *Latex Semantic Analysis* (LSA), *Hiper Space Analogue to Language* (HAL) e *Global Vectors* (GloVe). O segundo grupo de famílias de algoritmos são os métodos preditivos que tentam predizer as próximas palavras vizinhas dado o contexto (HARTMANN et al., 2017).

O trabalho de Bengio et al. (2003) apresentou um modelo de Aprendizado de Máquina para representar palavras mediante um vetor chamado *word embeddings*. Essa técnica permite a representação do texto, de acordo a seu significado com outras palavras semelhantes.

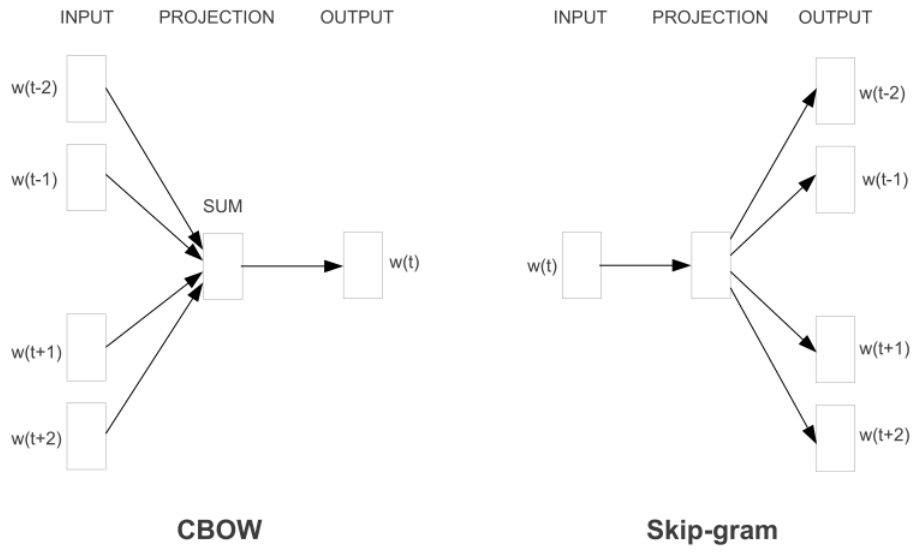
No entanto uma das maiores contribuições que popularizaram essa abordagem foi o *Word2Vec* (MIKOLOV et al., 2013a; MIKOLOV et al., 2013b) é um modelo de rede neural para a representação de *embeddings* ou sequência de palavras contínuas de um conjunto grande de dados. A ideia do *Word2Vec* é construir um vetor numérico de valor real, que representa uma palavra. O *Word2Vec* propõe duas arquiteturas de treinamento para a representação distribuída de palavras: *continuous bag of Words* (CBOW) e *skip gram*.

1. **CBOW**. O modelo *continuous bag of Words* utiliza uma representação contínua do contexto, onde a ordem das palavras na história não influencia sua projeção. O objetivo é prever a palavra do meio a partir do contexto. A arquitetura do modelo é apresentada na Figura 21.

2. **Skip-gram**. Esse modelo é oposto ao CBOW. No lugar de tentar predizer a próxima palavra baseado no contexto, o objetivo é prever o contexto de onde a palavra está inserida. Utiliza como entrada a palavra atual ao classificador *log* linear com uma camada de projeção contínua.

Esses modelos são ilustrados na Figura 21.

Figura 21 – Arquitetura distribuída do modelo *CBOW* (esquerda) e *Skip-gram* (direita), que prediz o contexto dado uma palavra.



Fonte: Mikolov et al. (2013a).

O trabalho de *Word2Vec*, trouxe a capacidade de aprender significados semânticos e úteis nas tarefas de PLN. Desse modo outras técnicas de vetorização de palavras foram

propostas. O *Global Vectors* (GloVe) (PENNINGTON; SOCHER; MANNING, 2014) para tarefas de analogias sintáticas e semânticas.

O *GloVe* é método consiste de uma matriz X de *word-word co-ocurrence* que é construída olhando o contexto das palavras. Cada posição da matriz X_{ij} contém a probabilidade de que uma palavra j esteja próxima a uma palavra i. Por exemplo, dado que $X_i = \sum_k X_{ik}$ é o número de vezes de qualquer palavra apareça no contexto da palavra i, finalmente a Equação 37 é a probabilidade da palavra j apareça no contexto da palavra i (PENNINGTON; SOCHER; MANNING, 2014):

$$P_{ij} = P(j|i) = X_{ij}/X_i \tag{37}$$

As *embeddings* geralmente são treinadas desde um corpus gigante não rotulado, o treinamento de uma *embedding* normalmente consume muito tempo, requer grande quantidade de memória, e uma GPU descente assim como altos requerimentos técnicos. Felizmente existem já *embeddings* pre-treinadas disponíveis para uso de qualquer tarefa de PLN. Porém dependendo da *embedding* elas contêm um contexto e vocabulário fixo, onde não reconhece novas palavras e ou outras *embeddings* contextuais. Por exemplo existem duas *embeddings* poderosas: *Facebook* oferece o modelo *fastText* e Google subministra o modelo *BERT* para várias línguas, que explicaremos na Seção 5.6.3.

Recentemente, o trabalho feito por Bojanowski et al. (2017) apresentaram o *FastText* é uma extensão da arquitetura Skip-gram do *Word2Vec*, onde cada palavra é representada como um saco de caracteres de n-grams dentro do modelo skip-gram. Essa característica lhe dá a capacidade de representar termos não observados no treino do "out of vocabulary" (OOV), lidar com sub palavras "subwords" uma característica de línguas com um vocabulário gigante e com palavras estranhas, além de capturar características morfológicas na formação dos termos por exemplo:

<eating> seria: 3-grams = <ea, eat, ati, tin, ing, ng>.

5.5 Rotulado Semissupervisionado

Como foi indicado na Seção 4.1.2 há casos onde existem muitos dados não rotulados no mundo real. Rotular um corpus é um processo custoso que leva tempo. Os exemplos rotulados são essências para o treinamento supervisionado, porém existem em poucas quantidades e pelo contrário existem abundantes exemplos não rotulados, dependendo do domínio a ser aplicado. Existem algumas técnicas na literatura que ajudam neste processo, a continuação explicaremos uma heurística simples mas poderosa.

5.5.1 Algoritmo Label Propagation

Proposto inicialmente por Zhu e Ghahramani (2002) e existe uma versão adaptada para a detecção de comunidades baseada em redes em (RAGHAVAN; ALBERT; KUMARA, 2007). Uma comunidade em uma rede é um grupo de nós que são semelhantes entre sim e dissimilares com o resto da rede.

O "Label Propagation" (LP) é um algoritmo iterativo que têm como essência que pontos no espaço muito próximos tendem a ter o mesma classe de rótulo. O LP é considerado um aprendizado transdutivo que utiliza um pequeno conjunto de dados rotulados junto com um grande conjunto de textos não rotulados durante o processo de aprendizado (ROSSI; LOPES; REZENDE, 2016), (SANTOS; ROSSI; MARCACINI, 2017). O LP resolve o problema da forma de propagação do rótulo, para isso os nós rotulados propagam para os nós vizinhos de uma rede de acordo a uma medida de proximidade, mantendo os nós rotulados sem modificatória, porém existem alguns frameworks que modificam o rótulo para melhora da rede de acordo a uma função de custo.

Segundo Subelj (2019) o LP é um algoritmo para a detecção de comunidades em redes, é um método de *clustering* e particionado simples e rápido. Para simplificar, descrevemos a estrutura básica de propagação de rótulos para o caso da detecção de comunidades em redes não direcionadas simples. Pode ser definido como um grupo de nós densos interconectados e não direcionado com n_i nós e Γ_i que é um conjunto vizinhos do nó $i \in \{1, 2, ...n\}$. Seja g_i a atribuição de grupo ou rótulo da comunidade do nó n_i a inferir.

A propagação do rótulo procede da seguinte maneira. Inicialmente, os nós são colocados em grupos separados atribuindo um rótulo exclusivo a cada nó como $g_i = i$. Então, os rótulos são propagados entre os nós até que o equilíbrio seja alcançado. A cada iteração de propagação do rótulo, cada nó i adota o rótulo compartilhado pela maioria de seus vizinhos Γ_i (SUBELJ, 2019).

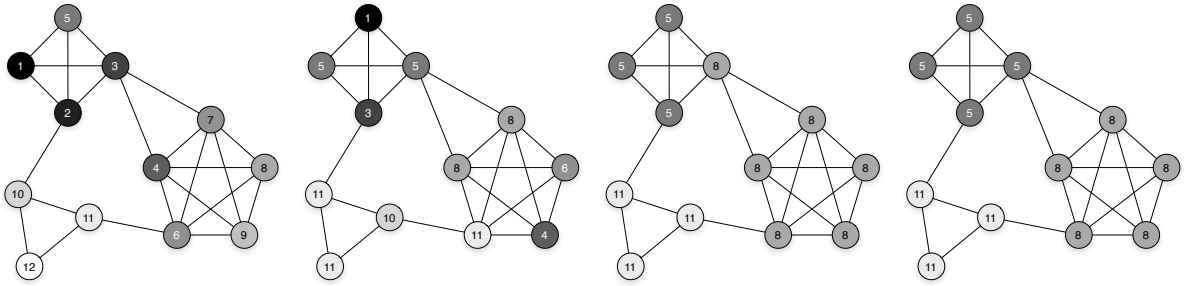
$$g_i = argmax|\{j \in \Gamma_i | g_j = g\}| \tag{38}$$

Após algumas iterações o algoritmo vai formando alguns grupos em áreas densas até chegar aos bordes da rede, a propagação converge quando a Equação 38 é valida para todos os nós e os rótulos não mudam mais, e grupos conectados de nós que compartilham o mesmo rótulo são classificados como comunidades. A Figura 22 mostra a propagação do rótulo com três iterações é capaz de detectar três comunidades

A complexidade do algoritmo esta dada por um comportamento linear para que seja completado, e depende do número de arestas da rede denotado por m. A cada iteração de LP, o rótulo do nó i pode ser atualizado com uma varredura no seus vizinhos que possui a complexidade $\mathcal{O}(k_i)$, onde k_i é o grau do nó i. Dado que $\sum_i k_i = 2m$, a complexidade de toda iteração do LP é $\mathcal{O}(m)$. Uma ordenação aleatória é realizada nos nós antes de cada iteração que toma $\mathcal{O}(n)$. Portanto, o tempo total da propagação de rótulo têm com-

plexidade = $\mathcal{O}(cn + cm)$, onde c é o número de iterações antes da convergência(SUBELJ, 2019).

Figura 22 – O algoritmo *Label Propagation* aplicado a uma rede, com três comunidades.



Fonte: Subelj (2019)

Esta heurística é uma solução ideal para o problema de textos não rotulados sem utilizar outros algoritmos não supervisionados de clustering (porém não é a única forma), pois permite representar a rede como um conjunto de nós e arestas que contém características geradas por algum tipo de algoritmo por exemplo: o *TF-IDF*, que nos permite espalhar os rótulos a partir de poucos textos rotulados por um supervisor chamados: como "seed" inicial de textos rotulados que espalham seus rótulos para nós ou textos sem rótulo semelhantes.

Uma parte importante para este algoritmo é o *input*, que contém uma rede com sua topologia com nós e arestas. Cada nó deste grafo representa um texto no espaço vetorial com n-dimensões. O modelo de espaço vetorial pode ser construído com o algoritmo *TF-IDF* ou melhor ainda com *Embeddings*. Para a criação das arestas entre nós deve existir algum tipo de critério de similaridade, que explicaremos na Seção seguinte.

Os algoritmos de classificação transdutiva em redes tem como base a função de regularização (frameworks), onde o objetivo é minimizar uma função de custo (SANTOS; ROSSI; MARCACINI, 2017). O alvo do algoritmo LP é minimizar iterativamente a função apresentada na Equação 39 até chegar a um ponto estacionário, objetivo de esses *frameworks* é encontrar um conjunto de rótulos que respeite duas condições:

a) As informações de classe dos objetos vizinhos devem ser semelhantes.

b) As informações de classe dos objetos rotulados atribuídos durante o processo de propagação devem ser semelhantes as informações de classes reais.

As condições anteriores podem ser inclusas em um *framework* de regularização, segundo o autor em Rossi (2015), a forma geral de uma função de regularização esta definida pela Equação 39:

$$\mathrm{Q(F)} = \frac{1}{2} \sum_{o_i, o_j \in O} w_{o_i, o_j} \Omega(f_{oi} - f_{oj})^2 + \mu \sum_{o_i, \in O^L} \Omega'(f_{oi} - y_{oi})^2 \tag{39}$$

Onde F é uma matriz de classe de todos os objetos, o_i é um objeto da rede, f_{oi} é o vetor com informação da classe do objeto o_i, O é um conjunto de todos os objetos da rede,

O^L é um subconjunto de objetos rotulados e y_{oi} é o vetor de informação da classe de um objeto rotulado. O termo Ω é responsável por calcular a similaridade entre os vetores de informação, o segundo termino Ω' calcula a proximidade entre informação de classe de objetos rotulados e suas correspondentes classes reais. O valor w indica o peso da relação entre objetos, e o valor μ indica a importância da informação de classe durante o processo de propagação de rótulos.

Um algoritmo de regularização importante é o algoritmo *Gaussian Fields and Harmonic Functions* (GFHF) (ZHU; GHAHRAMANI; LAFFERTY, 2003). O algoritmo GFHF é baseado em campos gaussianos e funções harmônicas. A principal característica é calcular o rótulo de um objeto não rotulado em base na média de informação do rótulo dos vizinhos ponderada pelos pesos das respectivas conexões. Outra característica importante do GFHF é que considera que o grafo é homogêneo e da muita importância a informação rotulada ou seja o rótulo não muda durante o processo de propagação, a função que minimiza está definida como (ROSSI, 2015):

$$\mathrm{Q(F)} = \frac{1}{2} \sum_{o_i, o_j \in O} w_{o_i,o_j} (f_{oi} - f_{oj})^2 + \lim_{\mu \to \infty} \mu \sum_{o_i \in O^L} (f_{oi} - y_{oi})^2 \tag{40}$$

Existem outro framework de regularização que permite em alguns cenários alterar os rótulos iniciais se precisar, chamado *Learning with Local and Global Consistency* (LLGC) (ZHOU et al., 2004), este *framework* de regularização em alguns casos podem alterar os rótulos inicias, porque pode existir objetos erroneamente rotulados, são muito ruidosos e com isso deteriorar a performance da classificação. A Equação 41 mostra seu cálculo (ROSSI, 2015).

$$\mathrm{Q(F)} = \frac{1}{2} \sum_{o_i, o_j \in O} w_{o_i,o_j} (f_{oi} - f_{oj})^2 + \mu \sum_{o_i \in O^L} (f_{oi} - y_{oi})^2 \tag{41}$$

O valor $\lim_{\mu \to \infty} \mu$ indica que o rótulo de um nó tem um peso muito importante na rede, ou seja a informação de rótulos nunca deve ser alterada, mantendo seus rótulos originais, para satisfazer essa restrição o valor μ é um valor grande. O termo $\frac{1}{2} \sum_{o_i, o_j \in O} w_{o_i,o_j} (f_{oi} - f_{oj})^2$ é uma função energia quadrática que tenta minimizar os vetores de informação entre os objetos rotulados na rede.

5.5.2 Medida de Similaridade

Em tarefas de agrupamento não supervisionado o objetivo é organizar um conjunto de objetos com algum tipo de medida de proximidade. A escolha de uma medida de similaridade entre objetos é fundamental para a construção de grupos de objetos, essa escolha depende do tipo de características dos dados.

Existem diferentes medidas de proximidade para dados discretos, dados contínuos ou a mistura, como: a distância Euclidiana, a distância Minskoski (FELDMAN; SANGER,

2006). A mais comumente utilizada para medir a similaridade entre dados textuais é a distância Cosseno.

A similaridade do Cosseno é definida pela Equação 42 (FELDMAN; SANGER, 2006; TAN; STEINBACH; KUMAR, 2006) como o ângulo cosseno formado entre os dois vetores de n-dimensões, considere dois documentos representados por : $x_i = (x_{i,1}, x_{i,2}, ...x_{i,n})$ e $x_j = (x_{j,1}, x_{j,2}, ...x_{j,n})$.

$$sim(x_i, x_j) = \frac{x_i.x_j}{||x_i||||x_j||} \tag{42}$$

Onde $||x_i||$ é a norma Euclidiana do vetor x_i definida como: $\sqrt{(x_{i,1}, x_{i,2}, ...x_{i,n})^2}$ similarmente se calcula para $||x_j||$. Esta medida calcula o cosseno do ângulo entre os vetores x_i e x_j. Se o valor do cosseno é 0 significa que o ângulo entre eles é 90 ou seja são ortogonais em consequência não existe coincidência. Porém, se o valor do cosseno do ângulo é perto a 1 então valores de ângulos entre dois vetores próximos a 0 terão coincidência alta entre textos.

5.5.3 Algoritmos Graph Embeddings

Existem algoritmos semissupervisionados de aprendizagem de características que são úteis para PLN. Estas são outras estratégias para rotular textos e baseiam-se em analisar um grafo. O estado da arte dos algoritmos de *Embeddings* de redes pode-se classificar em vários tipos, não é foco de nossa pesquisa aprofundar nesta área. Nesta seção só falaremos do algoritmo baseado em caminhos aleatórios (random walks do termo em inglês) e redes neuronais rasas.

Antes de falar de *network embeddings* é bom lembrar conceitos relacionados a *Embeddings*, especificamente o *Word2Vec* introduzido por (MIKOLOV et al., 2013a), estes conceitos foram explicados na seção 5.4.3.

O Word2Vec tem uma arquitetura SkipGram que foi o ponto de início para estes tipos de algoritmos, o trabalho recente na época e autores de Perozzi, Al-Rfou e Skiena (2014) trouxeram a ideia de enfatizar as semelhanças entre PLN e a análise de Redes Sociais. Existem analogias entre *Word Embeddings* e *Graph Embeddings*. Por exemplo se gera uma seqüência de nós, com caminhos aleatórios em uma rede social a freqüência de nós será igual a freqüência de palavras em um corpus, ambas distribuições seguem a lei Zip (PEROZZI; AL-RFOU; SKIENA, 2014).

O Node2Vec introduzido em Grover e Leskovec (2016) é um algoritmo para aprender representações contínuas para nós em redes, Node2Vec aprende a mapear nós a um espaço de baixa dimensão, maximizando a vizinhanças e preservando a topologia da rede. A principal idéia é fornecer sentencias de nós (em lugar de palavras) como entrada para os algoritmos *Word Embeddings* e a saída esta conformada por vetores de menor dimensão.

O Node2Vec pode ser utilizado com um algoritmo de clusterização como o KMeans (JAIN; DUBES, 1988) que agrupa dados de acordo a suas características de forma não supervisionada.

5.6 Algoritmo de Classificação

Nesta seção serão apresentados alguns conceitos relacionados a abordagens e algoritmos de classificação utilizados em nossa proposta. Assim como as métricas para avaliar a eles.

5.6.1 Abordagens para Classificação do Sentimento

Para a tarefa da classificação do sentimento relacionado ao aspecto existem duas principais abordagens. A primeira é utilizando Aprendizado de Máquina segundo e baseados em Léxicons (LIU, 2012; LIU, 2015):

1. **Algoritmos baseados em Aprendizado de Máquina**: Existem os algoritmos tradicionais para a classificação texto comumente utilizados do tipo Support Vector Machine (SVM), Naive Bayes (NB) as características utilizadas por esses algoritmos não são suficiente para a classificação do sentimento no nível da sentença e documento (LIU, 2015). A desvantagem dessas técnicas é que não considera a independência do alvo da opinião (uma entidade ou um aspecto) e são incapazes de saber a quem se refere. Para resolver esse problema foram adaptados os algoritmos para o alvo da opinião no processo de aprendizado, essa técnica utilizada foi utilizar um árvore de de *parsing* sintáctico para determinar a dependência.

   Esses algoritmos podem ser utilizados como *baseline* e o *performance*' depende muito do nível de complexidade do corpus (dataset), a tarefa de classificação, as características utilizadas (WANG; MANNING, 2012)

2. **Algoritmos Léxicons**: este tipo de abordagens faz uso de léxico de termos emocionais. Pode ser definida como uma estrutura altamente sistemática que define o significado das palavras e como elas podem ser usadas (SANTOS, 2019). Para (SPECIA; NUNES, 2004), os léxicos computacionais são recursos criados geralmente de forma manual, especificamente para o tratamento computacional. Em geral, uma abordagem léxica para avaliar o sentimento sobre aspectos usa os seguintes recursos (SUNDERMANN et al., 2019):

   - ❏ Um léxico de expressões de sentimento que inclui palavras, frases, expressões e regras de composição.
   - ❏ Um conjunto de regras para lidar com diferentes construções de linguagem (por exemplo, modificadores de sentimento e cláusulas) e tipos de frases.

- ❑ Uma função de agregação de sentimento ou um conjunto de sentimentos e relacionamentos de destino derivados da árvore sintática para determinar a orientação do sentimento em cada destino em uma frase.

As abordagens de PLN tradicionais se baseavam em uma análise segmentada (parsing) e em base a regras manualmente elaboradas por um supervisor. A natureza complexa, ambígua da linguagem e a necessidade de identificar a semântica e o contexto das palavras ultrapassam a capacidade humana em elaborar essas regras. Portanto, é desejável algoritmos capazes de resolver a estrutura sintática, desambiguações e compreender o escopo semântico de uma sentença.

Na próxima seção será explicado melhor como pode se aplicar os algoritmos de Aprendizado de Máquina usados neste trabalho. Não aprofundaremos nas abordagens Léxicos porque não é parte do escopo desta proposta.

5.6.2 Representações Contextuais

Como se explicou na seção anterior as arquiteturas relativamente recentes em PLN são muito importantes para representar as palavras num contexto distribuído. Até o presente momento temos visto representações estáticas para as palavras como o *Word2Vec* e o *GloVe*. As representações distribuídas estáticas de palavras possuem algumas limitações ao assumir um vetor único e universal para cada termo. Isso acarreta diferentes tipos de problemas como:

a) Desconsidera características essenciais da língua como a polissemia e homonomia, já que não é capaz de representar diferentes significados que uma mesma palavra pode assumir em diferentes contextos (NEELAKANTAN et al., 2015).

b) Mesma representação da palavra no mesmo contexto em que ela ocorre na sentença. Dependendo do contexto que aparece a palavra possui diferentes semânticas.

Por exemplo uma palavra pode ter diferentes sentidos e significados: “O banco não aceitará depósitos aos sábados” e “ O banco da praça está molhado”. O significado da palavra “banco” contém diferentes significados dependendo de seu contexto. Razoavelmente, pode existir duas representações vetoriais diferentes da palavra “banco” em base de seus dois significados diferentes. Essa foi a nova classe de modelos que têm essa capacidade de raciocínio divergindo do conceito de “global word representation” e propuseram “contextual word embeddings” (YOUNG et al., 2018).

O trabalho proposto pelos autores, Peters et al. (2018) o *Embedding from Language Model* (ELMo), foi considerado o “divisor de águas”, são embeddings contextuais, que produzem diferentes embeddings para cada contexto em que uma palavra é referenciada, assim consegue diferentes representações com diferente significado da palavra onde ela é usada. O ELMo marcou a diferença de modelos de linguagens bidirecionais *biLM*. O

ELMo consegue a partir de N diferentes sentenças onde a palavra w esta presente, gerar N diferentes reapresentações $w_1, w_2, w_3, ...w_N$. Adicionalmente, utiliza representações internas baseadas em caracteres, permitindo considerar características morfológicas dos termos e torna possível a representação de palavras fora do vocabulário de treinamento (GOMES; EVSUKOFF, 2019).

Paralelamente, outras propostas foram apresentadas baseados no modelo de linguagem, o *Language Model Fine-tuning for Text Classification* (ULMFit) (HOWARD; RUDER, 2018) um método efetivo para a transferência de aprendizado para qualquer tarefa de PLN, baseando-se em a transferência do aprendizado de *Computer Vision* (CV), estas técnicas são uteis para fazer o *fine tunning* de um modelo de linguagem.

Uma limitação nestes modelos de linguagem é que não consideram o contexto bidirecional (á esquerda e à direita) conseqüentemente os modelos são condicionados só a seu contexto anterior. Na próxima seção falaremos sobre modelos bidirecionais.

5.6.3 Representações Bidirecionais com Transformers: Bert

Um trabalho recente baseado em mecanismo de atenção *Bidirectional Encoder Representations from Transformers* (BERT) (DEVLIN et al., 2019) que estabeleceu novos parâmetros em estado-da-arte para diferentes *benchmark* e tarefas de PLN. o *Bert* é um dos modelos de linguagens mais exitosos baseado na arquitetura *Transformer* (VASWANI et al., 2017) e considera o contexto das palavras a direita e a esquerda em todas as camadas. Como resultado podem ser utilizadas para uma ampla gama de tarefas, ajustando apenas a última camada de saída para especializá-la sem fazer alguma modificação específica da arquitetura.

A maiorias dos modelos de linguagem são baseados em arquitetura unidirecional para aprender as representações gerais da língua, ou seja, as saídas são condicionadas apenas às palavras anteriores (do contexto esquerdo). Ao aplicar tais modelos em tarefas, os modelos estão limitados ao contexto esquerdo.

O BERT apresenta uma arquitetura de modelo de linguagem bidirecional para explorar esse conhecimento fazendo uso do *masked language model* (MLM). o MLM mascara cada *token* da entrada onde o alvo é prever o id do vocabulário original da palavra baseado só no seu contexto, desta forma é capaz de representar contexto de esquerda e direita (DEVLIN et al., 2019).

Uma das vantagens do BERT é que ele é capaz de transferir o conhecimento que tem no seu modelo pré-treinado, com uma base de dados pequena e especializá-la na última camada para um classificador binário ou multi-classe, assim reutilizar os pesos e camadas, isto é conhecido como *fine-tunning*.

Atualmente no momento de escrever esta dissertação existem outras variantes do BERT. O DistilBert Sanh et al. (2020) é uma versão destilada do BERT, solução mais rápida, barata e leve, no qual é possível reduzir o tamanho do modelo em 40%, man-

tendo 97% de suas capacidades de compreensão linguística e sendo 60% mais rápido no treinamento. O número de camadas é reduzido pela metade e as operações algébricas são otimizadas. O BERT precisa de um ambiente de hardware caro (memória e GPU) que às vezes não é acessível a muitos, o DistilBert é uma excelente opção mais barata.

5.6.4 Medidas de Avaliação

Após da escolha do modelo de Aprendizado de Máquina é sumamente necessário medir a capacidade de acertos ou erros para avaliar se o modelo consegue generalizar. Quando utilizamos estes modelos de classificação de sentimento, eles podem ser avaliados com as métricas tradicionais de qualquer classificação binária quanto múltipla.

Uma medida de desempenho utilizada na avaliação de um classificador $\hat{\mathbf{f}}$ é a sua taxa de erro, ou classificações incorretas.

Dado um conjunto de dados com n amostras, sobre qual serão feitas as avaliações, a taxa de erro é à proporção de objetos desse conjunto classificados incorretamente por $\hat{\mathbf{f}}$ que é obtida pela comparação da classe conhecida do exemplo x_i com rótulo y_i com a classe predita, $\hat{\mathbf{f}}(x_i)$ definida pela Equação 43 (FACELI et al., 2000).

$$erro(\hat{\mathbf{f}}) = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{\mathbf{f}}(x_i)) \tag{43}$$

Este tipo de medida equivale ao uso da função de custo entre $0-1$ Existe uma função custo associado aos rótulos obtidos pela função $\hat{\mathbf{f}}$.

A taxa de erro varia entre $0-1$, valores próximos a 0 são melhores. O complemento dessa taxa é a taxa de acurácia do classificador, definido pela Equação 44, neste caso valores próximos a 1 são considerados melhores (FACELI et al., 2000).

$$\text{accuracy}(\hat{\mathbf{f}}) = 1 - erro(\hat{\mathbf{f}}) \tag{44}$$

Existem várias medidas de acurácia para medir a capacidade preditiva do classificador. A matriz de confusão é uma ferramenta útil para à análise de quão bom é um classificador ou seja mostra o número de predições corretas e incorretas em cada classe, veja a Figura 23. O *True positive* e *True negative* diz-se se o classificador está correto, e o *False positive* e *False negative* diz-se que o classificador está classificando incorretamente (HAN; KAMBER; PEI, 2012).

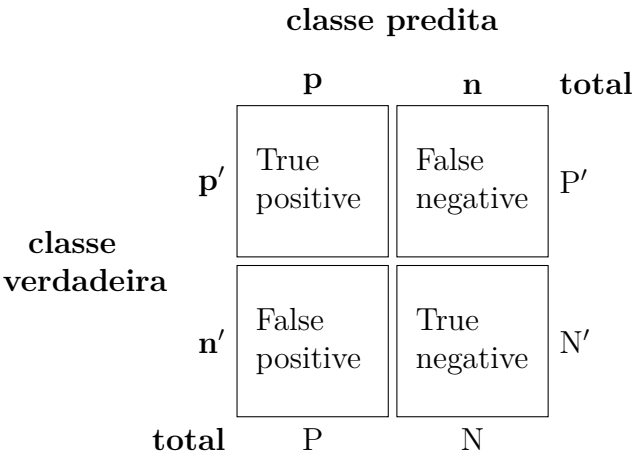


Figura 23 – Matriz de confusão para duas classes. Fonte própria.

As medidas que têm para avaliar a acurácia são:

Accuracy mostra o número de predições corretas e incorretas para cada classe.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} * 100\% \tag{45}$$

onde TP é *True Positive*; TN é *True Negative*; FP é *False Positive*; FN é *False Negative*.

F1-Score mede a qualidade da classificação evitando um bias das classes.

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{46}$$

As medidas *Precision* e *recall* também são usadas para problemas de classificação. A *Precision* pode ser pensada como uma medida de exatidão, enquanto *recall* é uma medida de completude (HAN; KAMBER; PEI, 2012). Estas medidas podem ser calculadas pelas Equações 47 e 48, assim:

$$precision = \frac{TP}{TP + FP} \tag{47}$$

$$recall = \frac{TP}{TP + FN} \tag{48}$$

Capítulo 6

Levantamento Bibliográfico

6.1 Metodologia da Pesquisa

Para realizar uma pesquisa eficiente é importante definir uma metodologia, sem ela é muito fácil sair dos objetivos propostos ou não chegar a ela.

Neste trabalho se optou pela revisão sistemática (RS). Para Kitchenham e Charters (2007), a RS tem como objetivo identificar, avaliar e interpretar toda a pesquisa disponível relevante para uma questão de pesquisa em particular, área temática, ou fenômeno de interesse. Esse processo está dividido em três principais fases Kitchenham e Charters (2007), Nakagawa et al. (2017): planejamento, condução e publicação dos resultados, conforme detalhados a seguir.

a) **Planejamento da revisão** tem como finalidade identificar a real necessidade, ou seja a motivação que levou a realizar a RS. Após de identificar, se define um protocolo que visa minimizar alguns tipos de vieses pelo pesquisador. Tal protocolo deve conter: as questões de pesquisa, a estratégia de busca, as fontes de pesquisa, as chaves de pesquisa, critérios de inclusão/exclusão.

b) **Condução da revisão** visa identificar estudos primários, selecionar estudos relevantes para a pesquisa, extrair e sintetizar os dados.

c) **Publicação dos resultados** refere-se a divulgar os resultados. Esta é a ultima fase que esta relacionada a escrita dos resultados que deveriam ser divulgados aos potenciais interessados por meio de relatórios, artigos de revista, conferencias para que sejam avaliados por a **revisão por pares**.

De acordo a estas sugestões a metodologia desta pesquisa segue os seguintes passos:

1. Estabelecimento dos motores de busca dos trabalhos relacionados como: *Periódicos Capes, ACM Digital Library, Scopus, IEEE Xplore, ElSeiver, Springer, Scopus, Google Scholar, Science Direct.* [1] para a pesquisa exploratória.

2. Definição das palavras-chave de procura como: "redes neurais artificiais *and* previsão de ações", "neural networks *and* time séries", "*forecasting*", "sentiment analysis for time séries", "deep learning *and* forecasting", "lstm *and* time séries forecasting"," *prediction assets*".

3. Busca do material referendado[2] por pares relevantes para nossa pesquisa como: periódicos, artigos, anais de eventos, teses e dissertações, livros. Não esta incluído material que, ao menos potencialmente, não passou pelo crivo de pares como *whitepapers, positions papers, blogs*, canais de *YouTube* de especialistas na área, vídeo aulas, Wikipedia, e monografias de conclusão do curso.

4. Seleção do material mais relevante de acordo as questões de pesquisa, ano de publicação e impacto, mensurado pelo número de referências.

5. Finalmente, se extrai os dados mais relevantes de cada artigo: como a metodologia usada, a tarefa a resolver, a hipótese a demonstrar e os experimentos realizados. Esse detalhamento é importante para fins de comparação e projeto do modelo neste trabalho.

6.2 Trabalhos Relacionados

Na literatura, encontra-se principalmente três principais abordagens para a previsão de séries temporais financeiras: os modelos estatísticos, modelos baseados em aprendizado de máquina e outro enfoque distinto baseado no processo de mineração de textos onde extraem conhecimentos de dados não estruturados como o texto, escrito em linguagem natural.

Nesta pesquisa se aborda os enfoques baseados em aprendizado de máquina especificamente o aprendizado profundo e a análise de sentimentos. Por isso, os modelos estatísticos e outros como a análise de sentimentos baseados em regras não serão foco de nossa proposta, porém abordamos com pouca profundidade.

O levantamento de trabalhos relacionados seguem uma sequência estabelecida para a explicação de cada um deles. Para cada trabalho se procuro encontrar: a) o problema que os autores tentaram resolver, b) há hipótese que os autores formularam para demonstrar, c) a metodologia ou técnicas que utilizadas para resolver o problema e d) os resultados que conseguiram demonstrar com a proposta.

[1] http://www.scopus.com, http://www.periodicos.capes.gov.br, https://www.sciencedirect.com, https://link.springer.com, https://ieeexplore.ieee.org, https://scholar.google.com.br/

[2] Aquele que teve o crivo de outros especialistas na área que avaliam uma dissertação ou publicação

6.2.1 Modelos baseados em Aprendizado de Máquina

Nesta seção, exploramos as pesquisas em métodos não paramétricos de Aprendizado de Máquina, ou seja, que não assumem conhecimento prévio da distribuição dos dados.

1. Parmezan e Batista (2015) pesquisam por modelos baseados em uma busca local não-paramétrica por similaridade, baseando a função de previsão com as k-sequências mais parecidas, em lugar de métodos globais que procuram por uma função preditiva baseada no treinamento dos dados. O problema identificado no trabalho deles foi, que os métodos baseados em previsão por similaridade falham na aplicação incorreta da invariância para evitar coincidências errôneas das k-sequências mais parecidas a uma consulta. A proposta consiste na modificação do algoritmo *k-Nearest Neighbor* (kNN) para demonstrar que os métodos por similaridade são capazes de prever o comportamento não linear das séries temporais.

   A hipótese levantada pelos autores foi, se ao adicionar três mudanças ao kNN, de modo a conseguir uma melhor precisão na previsão: a) a invariância a amplitude e deslocamento; b) invariância à complexidade; c) uma boa política para evitar os casamentos triviais. O novo algoritmo chama-se *k-Nearest Time séries Prediction with Invariances* (kNN-TSPI). Estas modificações consistem em:

   - *A invariância da amplitude e deslocamento*, refere-se por exemplo. Se há uma consulta Q (uma sequência de dados) e S^1, S^2 duas sequências mais parecidas a consulta Q, onde a Distância Euclidiana entre $DE(Q, S^1) = X_1$ e $DE(Q, S^2) = X_2$ e $X_1 > X_2$ mas S^2 é uma figura completamente diferente que S^1 que é muito parecida a consulta Q. Isto se deve a pequenos deslocamentos que causam um crescimento rápido na distância entre ambos objetos. Estas pequenas diferenças são suficientes para casamentos incorretos, já que poucas sequências se iniciam no mesmo valor.
   - *A necessidade da invariância à complexidade* refere-se à comparação entre sequências. Por exemplo dois objetos complexos, mesmo que similares, tendem a ter uma distância muito maior que objetos com simples. Ou seja, a distância Euclidiana entre $DE(Q, S^1) = X_1$ é maior que $DE(Q, S^2) = X_2$, onde S^2 é uma série simples, suave.

     A complexidade foi incorporada com seis diferentes medidas que utilizam conceitos de *teoria do caos* e *teoria da informação*.
   - *Casamentos triviais* refere-se ao casamento de sequências muito próximas no eixo do tempo. Por exemplo, uma janela de busca que inicia em m pode ter sequências muito parecidas que compartem muitas características a outras se somente deslocamos $m+1$ ou $m-1$, o que não traz informações úteis à predição.

Para solucionar esse problema, os autores sugerem incorporar diversidade na consulta das k subsequências mais similares.

Os resultados mostram que, ao adicionar o tratamento da invariância à complexidade, a previsão baseada na similaridade melhora. Realizarem-se 3 experimentos, o kNN-TSPI foi comparado com dois modelos, o *Multilayer Perceptron* (MLP) e *Support Vector Machine* (SVM), onde encontraram que ao utilizar a distância da complexidade da invariância optem uma melhora na acurácia com o MLP mas é superado com o SVM. Foram usadas um *dataset* de 55 séries temporais de diferentes ramos e disponibilizarão em um servidor no ICMC da USP.

2. No trabalho de Teixeira e Oliveira (2010), propõe-se usar um modelo híbrido fazendo uso de indicadores técnicos e um classificador simples k-NN para prever o preço da tendência diária. O uso de um classificador simples combinado com quatro indicadores técnicos que a academia usualmente ignora como: *Simple Moving Average* (SMA), *Relative Strength Index* (RSI), *Stochastics* propostos por Murphy (1999). Utilizando os valores dos preços e volume diário junto com ferramentas do *trading*, como *stop loss* e *stop gain*, teve bons resultados em termos de rentabilidade em comparação de uma estratégia *buy-and-hold* contra outros classificadores.

   A metodologia adotada nesse trabalho primeiramente realiza um experimento com uma base de dados de uma corretora brasileira com dados dos anos $1998 - 2009$ e com um balance inicial. Utiliza 22 características como entrada ao algoritmo, entre elas: $x(t)$, $x_{high}(t)$, $x_{low}(t)$, $v(t)$ (preço de fechamento, preço mais alto, preço mais baixo e volume), PMA_s e PMA_l (*short-term* e *long-term* são as medias moveis aplicadas aos preços), VMA_s e VMA_l(*short-term* e *long-term* são as médias móveis aplicadas ao volume). Segundo, os dados são divididos em 10 subconjuntos, onde pega 3 subconjuntos para treino e 1 para teste e depois desloca 1 ano e pega outros 3 para teste e 1 para treino.

   Em quanto ao modelo, este contempla algumas técnicas dos *traders* como o *stop loss* e *stop gain* explicadas na Secção 2.1.1 para cuidar dos investimentos e configura uma perda ou risco do 3% e 10% de ganho nas operações de compra e venda. Por exemplo, se o modelo prediz comprar e o preço cai por debaixo do 3% do risco ou 10% todas as ações são vendidas, além disso adiciona ao modelo um filtro *RSI* que serve para cuidar ainda mais os investimentos. Por exemplo se o modelo prediz comprar e o preço é maior que $RSI(price(t)) > 70$, que significa se o preço esta por acima do RSI permitido de 70 então a operação de compra tem muito risco porque esta sobre comprado o ativo então o sistema ignora essa recomendação.

   Os resultados dos autores, concluíram que ao usar um classificador simples k-NN junto a estratégias e conceitos simples como o *stop loss*, o *stop gain* e o filtro *RSI*

conseguiu resultados melhores em termo de lucros comparado com uma estratégia de *buy-and-hold*, em lugar de utilizar classificadores complexos treinados com algoritmos genéticos ou outras redes neurais.

3. O trabalho de Fischer e Krauss (2017) enfoca-se na problemática da previsão dos movimentos das ações em séries temporais utilizando *Deep learning* (DL). A hipótese que demonstra é a superioridade do DL sobe outras técnicas, especificamente os autores utilizam uma variação das Redes Neurais Recorrentes chamada *Long Short Term Memory* (LSTM), uma técnica para aprendizado de sequências com memoria, em lugar de outras técnicas sem memória como o *Random Forest* (RF), *Deep Neural Network* (DNN) especificamente uma (Feed fordward Neural Network FFNN) e *Logistic Regressor* (LOG).

   A metodologia usada consistiu em experimentos com os modelos. Se identificou os seguintes passos. Primeiro, utiliza uma base com grande volume de dados entre $1990 - 2015$ de *Thonsom Reuters*, com 500 índices de ações do *S&P* 500. Segundo, divide os dados para treino e teste, o período de treino é de 750 dias e o teste é de 250 dias; em total 1000 de um conjunto de 500 ações n_i. Terceiro, define as características para a rede LSTM, conformada pelo preço de uma ação s representado com $P^s = (P^s_t)_{t \in T}$ no tempo t onde $s \in \{1 \dots n_i\}$ e define as séries temporais para cada n_i composta pelos retornos do preço, veja a Equação 49.

$$R^{m,s}_t = \frac{P^s_t}{P^s_{t-m}} - 1 \tag{49}$$

   A classificação para cada ação é definida pela variável Y^s_{t+1} que classifica binariamente, dependendo do valor $R^{1,s}_{t+1}$ no período $t+1$, se é menor que a média de todos os retornos no período $t+1$ é 0 e se fosse maior é 1. Enquanto a arquitetura foi conformada por uma característica de 240 sequências, uma camada oculta de $h = 25$ neurônios e a camada de saída tem 2 neurônios com uma função de ativação *softmax*.

   A previsão é feita pela probabilidade do preço $P^s_{t+1|t}$ de todas as ações s que estão abaixo ou acima da média no tempo $t+1$ utilizando só informação até o tempo t. Depois, faz-se um ranking de ordem descendente com as melhores classificações das ações. A que está no topo do ranking corresponde à mais valorizada que se espera que tenha uma melhor performance que a média no tempo $t+1$ e dependendo disso ele vai a *long*[3] no topo da lista ou *short* no último da lista.

   Os resultados da rede LSTM superam os métodos sem memória, com retornos estatisticamente e economicamente significativos de $0,46\%$ por dia - comparado a $0,43\%$ com RF, $0,32\%$ com o DNN e $0,26\%$ com a LOG.

[3] "Long" e "short" são utilizados como sinônimo de "comprar" e "vender" quando se opera nas bolsas de ações

4. Em Laptev et al. (2017), os autores descrevem a problemática da previsão de "eventos extremos" nas séries temporais dependem de vários fatores externos como o clima, crescimento na população, câmbio nas vendas. O modelado de estas variáveis exógenas (variáveis que afetam o modelo mas estão fora como o clima) e a extração de características em dados gigantes representa uma tarefa difícil. O artigo apresenta uma nova arquitetura como proposta para a previsão de séries temporais que inclui o modelado de variáveis exógenas, este modelo é baseada em um *Autoencoder* para a extração de características em combinação do uma rede LSTM pela capacidade de modelar características não lineares. O modelo é baseado em:

   - ❏ A estimativa da incerteza, o problema de eventos extremos é um problema estatístico por natureza. Os autores combinam duas abordagens *BootStrap* e as redes Bayesianas para produzir um limite simples, robusto e restrito, com boa cobertura e propriedades de convergência baseados em trabalhos de Gal (2016) e Kendall e Gal (2017).
   - ❏ Utiliza um extrator de características via um *Autoencoder* que produz um vetor de características menor para ser concatenado a nova entrada para outro preditor LSTM.

   Os resultados obtidos estão baseados, Primeiro no cálculo das corridas do *Uber* feitas no EUA, que prevê os eventos extremos de corridas em dias especiais como: Natal, dias de férias, ano novo, dia da independência etc. Segundo, nas séries temporais em geral. A métrica usada para medir o erro nas previsões foi SMAPE, onde conseguiu um $2\% - 18\%$ de acurácia. A série em geral foi comparada com o erro do *dataset* público do *M3* Makridakis e Hibon (2000) .

5. O trabalho de Bao, Yue e Rao (2017), apresentou um novo modelo pouco usado mas com bons resultados baseado em Aprendizado não Supervisionado e Profundo. As técnicas de aprendizado profundo geralmente utilizadas são *Convolutional Neural Networks* (CNN), *Deep Belief Network* (DBN) e *Stacked Autoencoders* (SAE). Os autores ressaltam a falta de uso de SAE na previsão das séries temporais financeiras junto com as redes profundas para aprendizado de sequências com a LSTM. A hipótese que eles definiram foi, se a entrada da primeira camada de uma rede LSTM, aplica-se antes um método não supervisionado SAE que consiga aprender características mais compactas e que junto com uma transformada Wavelet Transform (WT) que consiga gerenciar a não estacionariedade e padrões irregulares nas séries temporais fosse utilizada como entrada no classificador de sequências a LSTM poderia melhorar a acurácia e rentabilidade das séries temporais.

   O modelo propostos pelos autores consistem de tries partes: transformações Wavelets, aplicação de um método não supervisionado SAE e o treino da rede LSTM:

- ❏ Primeiro, o modelo utiliza uma técnica do processamento de sinais para eliminar o ruído presente nas séries temporais com um método discreto chamado *Haar Wavelets* com a habilidade de gerenciar a não estacionaridade e padrões não regulares nos dados. As WT são funções matemáticas que descompõem o dado em diferentes frequências ou escalas para que consiga distinguir características do mundo real presente nas séries temporais como a sazonalidade e a volatilidade, em lugar de uma suposição de estacionaridade (HSIEH; HSIAO; YEH, 2011). A WT também é usada para gerenciar os padrões de alta irregularidade e reduzir o *overfitting*.
- ❏ Segundo, o SAE de jeito não supervisionado aprende características abstratas para gerar um vetor compacto destas. Nesta parte utiliza como entrada as seguintes características: os preços, volume do ativo, indicadores técnicos e indicadores macroeconômicos em total 19 características.
- ❏ Terceiro, treinar a LSTM para prever o valor futuro do preço.

Discutimos as medições de desempenho nesta parte. Primeiro demonstramos as medidas de precisão selecionadas para julgar o desempenho preditivo. Em seguida, discutimos como testamos o desempenho de rentabilidade de cada modelo.

Para o desempenho do modelo foram avaliados com duas métricas: a acurácia e a rentabilidade. Primeiro, para medir à acurácia da previsão foi utilizada a *Mean Absolute Percentage Error* (MAPE), *Correlation Coefficient* (R), *Theil's inequality Coefficient* (Theil U). Segundo, para avaliar a rentabilidade foi desenvolvida uma estratégia de *buy-and-sell* baseada nos resultados previstos pelo modelo e foram comparados com os retornos de uma estratégia *buy-and-hold* para cada ação. Esta avaliação é muito importante já que os autores procuram modelos que permitam as maiores rentabilidades além da acurácia.

Para avaliar a performance do modelo WSAE-LSTM proposto, este foi comparado com três modelos: LSTM, RNN, WLSTM. Os dados utilizados foram de seis mercados durante um período de seis anos: CSI 300 da China, Nifty 50 da Índia, Hang Seng índice do Hong Kong, NIkkei 225 de Tokyo, S&P 500 e DJIA de New York. A WSAE-LSTM tem uma taxa menor de error MAPE de 0.019 de CSI 300 e uma alta acurácia na rentabilidade meia de 40% sobe outros modelos onde demonstra a superioridade de seu modelo.

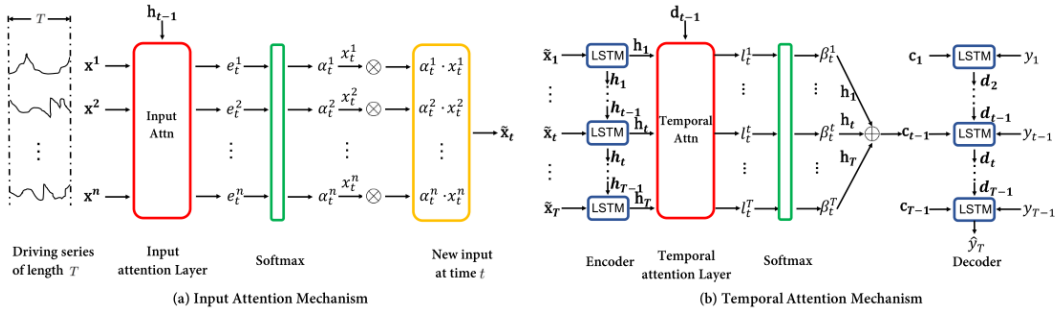
6. Os autores em Qin et al. (2017), declaram que a problemática no momento de prever os preços do mercado financeiro é a não linearidade das séries temporais e que é difícil capturar esses comportamentos em modelos estatísticos como o *ARIMA*. A academia tem realizado esforços para fazer previsões via métodos: *kernel Radial Base Function* (RBF) em Chen, Wang e Harris (2008), técnicas baseadas em *ensemble* Bouchachia e

Bouchachia (2008) e processos Gaussianos Frigola e Rasmussen (2014). O problema nestos abordagens radica que não conseguem capturar as verdadeiras relações não lineares escondidas devido a que usam formas não lineares pré-definidas.

Assim, a proposta dos autores está baseada na teoria de atenção humana, que diz-se que os comportamentos são melhor modelados em um mecanismo com doble atenção. A primeira fase, refere-se a selecionar as características que servem de estimulo. A segunda fase, utiliza informação categórica para decodificar o estimulo. Para implementar esse modelo, baseia-se em uma derivada das *Recurrent Neural Network* (RNN), a LSTM. Propõe um mecanismo com doble atenção e a chama *Dual Attention based Recurrent Neural Model* (DA-RNN) veja a Figura 24. O modelo contém um mecanismo de atenção (*encoder*)para extrair as características mais relevantes das séries, na segunda parte contém um mecanismo de atenção temporal para selecionar as características codificadas e capturar com um modelo de dependência longa.

A metodologia adotada pelos autores está dividida em duas fases, como ilustrado na Figura 24. Primeiro, com um *encoder*, que basicamente é uma LSTM que codifica as séries mais relevantes mediante um mecanismo de atenção nas entradas $X = \{x_1, x_2, x_3, ..., x_t\}$, com o alvo de aprender um mapeamento de x_t a h_t, onde $h_t = f(h_{t-1}, x_t)$ onde h_t é um estado oculto do codificador. Segundo, no *decoder*, desensolvem um mecanismo de atenção temporal utilizado para selecionar os estados ocultos mais relevantes. Esse decodificador é basicamente uma LSTM que decodifica as entradas codificadas. Com esse mecanismo consegue capturar as séries mais relevantes na entrada de dados e capturar as dependências temporais de longo prazo da entrada codificada.

Figura 24 – Ilustração da DA-RNN com dois mecanismo de atenção.



Fonte: Qin et al. (2017)

Os resultados que conseguiram foram com dois experimentos. Utilizaram dois bases de dados de SML 2010 e NASDAQ 100 para fazer as provas e as métricas utilizadas de comparação foram: RMS, MAE, MAPE. O treino foi realizado como um algoritmo *Stochastic Gradient Descent* (SGD). Utilizaram quatro modelos para

comparar o ARIMA, RNN, Encoder-Ddecoder Network, Attention-RNN. Os autores mostram que a DA-RRN tem melhor acurácia que as anteriores propostas, cabe mencionar que ainda tem melhor rendimento que a Attention-RNN que somente presta atenção em uma entrada.

6.2.2 Abordagem baseados em Análise de Sentimentos

Existem principalmente na literatura dois abordagens para realizar a tarefa de análise de sentimentos nos textos de língua humana. Existem métodos que utilizam o aprendizado de máquina e outras estão baseados em Léxico.

Dito isso nesta parte de trabalhos relacionados procura-se trabalhos com ênfases em aprendizado de máquina, os abordagens baseados em léxico não são parte do escopo de nossa pesquisa, porém serão mencionados como o caso dos autores Hutto e Gilbert (2014).

1. (DING et al., 2015) faz uso de uma rede CNN para modelar a influencia de eventos short-term e long-term que afetam ao mercado com o alvo de prever os movimentos do preço do mercado S&P500. O alvo desta pesquisa é aprender *embeddings* para eventos estruturados.

   O método utilizados é baseado numa Neural Tensor Network onde a entrada da rede é um conjunto de *embeddings* e a saída são eventos *embeddings*. O modelo de previsão é baseado em eventos que aconteceram no més, semana e dia anterior para prever eventos long-term, short-term e mid-term. Para aprender utiliza uma red CNN e a saida é uma das classes $1, -1$ que diz que o preço de uma ação vai subir ou cair

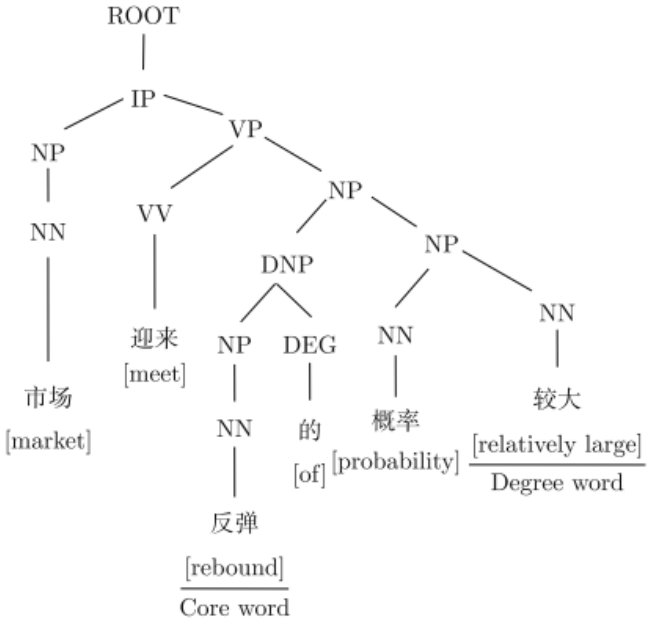
2. Para Wang (2017), o uso da análise qualitativa ou sentimental nos textos é muito importante. Para eles, há muito tempo o mercado financeiro utiliza a análise de séries temporais, olhando apenas o passado, mas deviera-se considerar os eventos que acontecem ao redor. Além disso, destacam que as tendências de índices futuros não só estão relacionados ao passado, mas também a opiniões públicas, eventos, parcerias e inclusive, opiniões de especialistas, que fazem mudar as tendências. Para o autor, é muito difícil conseguir modelar e prever o futuro somente baseado na análise do passado. Ele considera fatores que influenciem o presente, com o alvo de melhorar o modelo das séries temporais.

   A hipótese que demonstra é que, ao incorporar um indicador (informação de sentimento) que seja o resultado do processamento de *tweets*, textos financeiros e opiniões públicas em um modelo da rede neural como o *Nonlinear Autoregressive Exogenous* (NARX) então conseguirá fazer uma previsão dos índices de ações com melhor acurácia.

A metodologia nessa proposta baseia-se em experimentos e utiliza duas séries temporais para a previsão. Primeiro, utilizou uma série temporal com os preços de fechamento do índice da ação *Shangai Stock Exchange Composite Index* (SSECI) do ano 2012. Segundo, constrói-se uma série temporal de sentimentos fazendo a análise de sentimento em textos colhidos de microblogs em língua Chinesa. Para o processamento dos textos se realizou uma segmentação dos textos, dividindo cada texto do microblog em um conjunto sentença **s** e cada **s** em um conjunto de palavras **W**.

Para analisar a sentencia se construiu um árvore gramatico com *parser* (*Standford parser*), utilizado muito em tarefas de Processamento de linguagem Natural (PLN) para classificar se uma palavra é um *NN, NP, VP noun, noun phrase, verb phrase* (dos termos em Inglês) respectivamente. Veja a ilustração seguinte na Figura 25.

Figura 25 – Resultado do *parsing* na sentencias que diz a que classe gramatical pertence uma palavra.



Fonte: Wang (2017).

O algoritmo de análise de sentimentos procura as palavras mais importante de uma sentença e faz um contagem sofisticado para ver o grau de frequência, o qual expressa o peso do sentimento. Após de fazer o análise das sentencias, constrói a série temporal que está conformada pelo peso do sentimento w_F e é incorporado no modelo NARX, veja a Equação 50.

$$I_t = NARX_H(w_{F_{t-1}} \dots w_{F_{t-d}}, I_{t-1} \dots I_{t-d}) \tag{50}$$

Onde I é o valor de um índice da ação, t é a data do valor, H é o número de neurônios ocultos e w_F é o peso do sentimento.

Os resultados foram comparados com o algoritmo *TFIDF* e a métrica MSE. Em alguns casos, têm melhor comportamento e em outros não. O autor conclui falando em trabalhos futuros que pode melhorar a acurácia do modelo de sentimento agregando um dicionário de pesos de palavras e que a língua chinesa é muita restrita para outros domínios.

3. O trabalho dos autores Hutto e Gilbert (2014) também enfocaram na problemática d análise de sentimentos e opiniões de textos nas redes sociais. O grande volume de dados públicos gerados no *Twitter* e no *Facebook* representam sérios desafios em aplicações práticas da análise de sentimentos que precisam de um tratamento computacional para medir a subjetividade. Os autores expõem a necessidade de representar um modelo baseado em regras gerais para a análise de sentimento para redes sociais sensível à "polaridade" e "intensidade" do sentimento, sem utilizar modelos baseados em aprendizado de máquina, devido a que precisam um treinamento e rotulado dos dados extra.

   A hipótese do trabalho dos autores, esta relacionada à validação do modelo *Valence Aware Dictionary and Sentiment Reasoner* (VADER) proposto, um léxico de sentimento baseado em regras simples capazes de construir mecanismos de análise de opinião, usando a combinação de métodos qualitativos e quantitativos que ao usar melhorariam muito a acurácia da análise de sentimento sem a necessidade de um extenso processo de aprendizado e treino que fazem os modelos atuais de Aprendizado Supervisionado.

   O método baseia-se no desenvolvimento e validação de um "léxico de sentimento" de padrão sensível à polaridade e à intensidade de sentimentos expressos em *microblogs* das mídias sociais. Para realizar isto, utilizou a validação por humanos (apesar de ser uma atividade muito trabalhosa, intensa e propensa a erros), com bancos de dados de sentimentos conhecidos *Linguistic Inquiry Word Count* (LIWC), *General Iquiry* (GI), *Affective Norms for English Words* (ANEW) já prontos, que agrega características léxicas de sentimento como: *emoticons* **":-)"** – carinha feliz que expressa um sentimento positivo – e acrônimos (como LOL e WTF) chegando a reunir cerca de 7500 léxicos. Uma vez identificadas as características, avalia a intensidade do sentimento com uma abordagem do tipo *wisdom-of-the-crowd* (WotC) para adquirir uma estimativa do sentimento (intensidade) de cada uma das características lexicais. Com ajuda de avaliadores humanos independentes, conseguiu mais de 90.000 avaliações. A intensidade do sentimento no texto é expressado a cordo à normalização do valor entre (-1,1):

$$sentimento = \begin{cases} \text{se} \quad 0,6 \leq x \leq 1 & \text{muito positivo} \\ \text{se} \quad 0,2 \leq x < 0,6 & \textit{positivo} \\ \text{se} \quad x = 0 & \textit{neutro} \\ \text{se} \quad -0,6 \leq x < -0,2 & \textit{negativo} \\ \text{se} \quad -1 \leq x < -0,6 & \text{muito negativo} \end{cases}$$

O trabalho utiliza avaliadores humanos pré-selecionados, para o idioma inglês e com compreensão de leitura oferecendo um incentivo por escolher cada característica bem para conseguir melhor média e desvio do padrão. Define heurísticas para identificar propriedades e características em textos(*tweets*), pegando 400 positivos e 400 negativos de dados públicos no *Twitter* baseando-se na pontuação de Pattern.en [4] (que é uma ferramenta para pontuar sentimentos de acordo a adjetivos ingleses).

O objetivo de fazer isso é identificar mudanças na intensidade do sentimento nos textos, além do que normalmente seria capturado em um modelo *bag-of-words* com relações de ordem nos textos: a) A sinal "!" incrementa a pontuação sem modificar a ordem semântica, como por exemplo: "The food here is good" tem menos intensidade do que "The food here is good!" b) As palavras em maiúsculas são mais intensas por exemplo: "The food here is GREAT!" é mas intenso do que "The food here is great!", sem modificar a semântica. c) Identifica os modificadores ou advérbios por exemplo "The service here is extremely good" é diferente de "The service here is good" ou "The service here is marginally good", que reduz a intensidade. d) Identifica a conjunção "but", que sinaliza uma polaridade neutra "The food here is great, but the service is horrible".

Os resultados do VADER foram baseandose na comparação com analisadores sentimentais léxicos muito consolidados nesta área, como o LIWC, GI, ANEW, SentiWordNet (SWN), SenticNet (SCN). A classificação foi multi classe (positivo, neutro, negativo) e a avaliação é feita por meio da métrica *F1 score*. Os autores mostram um coeficiente de correlação do VADER ($r = 0,881$) que tem bom performance, melhor que os avaliadores humanos individuais ($r = 0,888$) de um grupo de 20 avaliadores humanos para a medir intensidade do sentimento de cada *tweet*. A precisão da classificação do *VADER*, com a métrica F1 score é $F1 = 0,96$, o que supera os avaliadores humanos individuais ($F1 = 0,84$) ao classificar corretamente o sentimento dos *tweets* em classe positiva, neutra ou negativa. O *VADER* conserva o benefício de tradicionais léxicos sentimentais como o LIWC mas é mais sensitivo as expressões de sentimentos em contexto sociais.

[4] https://www.clips.uantwerpen.be/pages/pattern-en#sentiment

O *VADER* tem melhor desempenho em comparação a técnicas de aprendizado de máquina treinadas no mesmo contexto e domínio. Os autores, conseguem demonstrar que um modelo baseado em regras e heurísticas com regras accessíveis têm melhor desempenho que um modelo de caixa preta.

4. O trabalho de Kim et al. (2016), resolve o problema da análise de sentimentos em comentários de algumas comunidades importantes e com maior capitalização no entorno de cripto-moedas. Os autores consideram que esses espaços poderiam conter opiniões que afetam as flutuações dos preços. Ele define como hipótese que os comentários e respostas de usuários nos fóruns do *Bitcoin*, *Ripple* e *Ethereum* (cripto-moedas) podariam afetar nas flutuações dos preços e o volume negociado diário.

   O método proposto para fazer a previsão, primeiro baseia-se em rotular os comentários com o algoritmo de Hutto e Gilbert (2014), que é um modelo baseado em regras para a análise de sentimentos, que mede a força ou intensidade do sentimento em um texto e categoriza entre um valor continuo $(-1, 1)$. Veja no trabalho anterior os valores da intensidade do sentimento.

   Segundo, realiza uma análise de associação entre o resultado das opiniões e as flutuações dos preços. Esta comparação foi feita pandronizando-se as duas séries temporais(uma série com preços e uma série com o resultados do analise da opinião) com o *z-score* (que transforma o sinal tal que ele possua média igual a 0 e desvio padrão 1 em uma janela de tempo) e aplicando o test de causalidade *Granger causality* (GT) criado por Granger (1969) e usado por Kim et al. (2016), Bollen, Mao e Zeng (2011) para ver se existe algum tipo de causalidade ou relação entre ambas séries temporais. O GT se baseia na suposição de que, se uma variável X causa Y, então as mudanças em X ocorrerão sistematicamente antes de mudanças em Y (KIM et al., 2016). Após aplicar a padronização, se construiu dois modelos com os valores desfasados do preço $S_t = S_t - S_{t-1}$ (do dia atual e anterior) e outra série com os valores desfasados $X_t = X_t - X_{t-1}$ do sentimento. Os modelos de séries estão expressados assim:

$$M1 : S_t = \alpha + \sum_{i=1}^{n} \gamma_i S_{t-i} + \epsilon_t \tag{51}$$

$$M2 : S_t = \alpha + \sum_{i=1}^{n} \gamma_i S_{t-i} + + \sum_{i=1}^{n} \beta_i X_{t-i} + \epsilon_t$$

   Após aplicar o teste, se aceita ou rejeita a algumas das hipóteses nula H1 (os comentários não afetam aos preços) ou alternativa H0 (os comentários sim afetam ao

preço). Se $H0 : \beta_1 = \beta_2 = ... = \beta_3 = 0$ então X_t não causa ou afeta a S_t e se ao menos existe um $H1 : \beta_1 \neq \beta_2 \neq ...\beta_3 \neq 0$ aceita a hipótese alternativa.

Os resultados contém dados das três cripto-moedas ao redor de um ano. Os resultados da associação têm o valor do *p-values* < 0.005 de 13 dias que diz que a hipótese alternativa foi aceita ou sim existem relação entre os preços e os comentários. Para medir a precisão dos modelos utilizo o F-measure e o coeficiente de *correlation Matthews* (MCC). O *bitcoin* utilizo 793 dias, onde dividiu 88% para aprender e 12% para teste e encontrou muita associação com os comentários positivos com 79% de precisão.

6.2.3 Modelos Estatísticos

Durante os últimos trinta anos, para resolver o problema da predição dos preços foram comumente utilizados os modelos lineares estatísticos como *Exponential Smoothing*, o bem conhecido e amplamente utilizado modelo *Autoregressive Integrated Moving Average*(ARIMA) (ZHANG, 2003) e modelos *Random Walk* (RW).

1. O estudo comparativo realizado por Parmezan, Souza e Batista (2019) mostra os benefícios e limitações dos algoritmos estatísticos e as técnicas de aprendizado de máquina para a previsão de séries temporais. O problema que detectaram os autores é que não existem na literatura pesquisas robustas que comparem os métodos paramétricos baseados em *Autoregression* e *Moving Average* (MA) e as técnicas não paramétricas de Aprendizado de máquina. Os métodos estatísticos assumem que os dados têm uma **distribuição a priori** conhecida que é usada como parâmetro para construir o modelo de previsão, porém esses parâmetros precisam de um conhecimento profundo dos conceitos matemáticos e experiencia técnica para modelar a função Parmezan, Souza e Batista (2019).

   Por outro lado existem os métodos não paramétricos de Aprendizado de Máquina **sem o conhecimento a priori** da distribuição dos dados, o qual é uma das principais vantagens dessa abordagem já que não pressupõe a natureza da distribuição de deles.

   O aporte ao estado do arte foi demostrar quais algoritmos são os melhores para ser utilizados na previsão das séries temporais, além da construção de um repositório de *datasets* de *benchmark* com dados reais e sintéticos para que outros pesquisadores façam experimentos com eles. Os autores realizaram uma revisão sistemática de 117 artigos e encontraram que as abordagens mais usadas foram 54% não paramétricos, 25% paramétricos e 21% híbridos. Os métodos mais usados foram Artificial Neural Network (ANN), Autogressive Integred Moving Average (ARIMA), Support Vector Machine (SVM), Hybrid, k-Neardest Neighbors (kNN), Fuzzy Logic (FL), Deep

Learning (DL), Bayesian Neural Networks (BNN), Simple Exponencital Smooting (SES), Wavelt Transform (WT), Holts Winters (HW) and Gausian Process (GP)

A partir disso, os autores selecionaram os algoritmos: SARIMA, ARIMA, SVM, LSTM, kNN-TSPI, MLP para avaliar. Foram utilizados 40 *dataset* sintéticos e 55 dados de domínios reais (agricultura, temperatura, engenhe-iria, climatologia, medicina, finanças, turismo etc) com características de não estacionariedade, crescimento de tendência, sazonalidade etc. Foram utilizadas as técnicas comuns para encontrar os valores paramétricos dos algoritmos como: *Cross Validation* (que procura os valores l, C termo de regularização e τ distribuição gaussiana do kernel), *Holdout Validation* (que procura os valores ideais para k e l para um algoritmo kNN) e *Box-Jenkins* para o modelo ARIMA e SARIMA. As medidas de acurácia usadas foram: *Mean Square Error* (MSE), Theils U Coefficient (TU) e *Prediction of Change of Direction* (POCID), devido a que usar vários critérios para avaliar um modelo é muito difícil os autores decidiram misturar as anteriores medidas em *Multi-Criteria Performance Measure* (MCPM) onde se minimiza o MSE e o TU tem que se maximizar o POCID.

Os resultados foram divididos em estudos comparativos: a) modelos preditivos aplicados a dados sintéticos com séries caóticas, determinísticas e estocásticos. b) modelos preditivos aplicados a dados reais. c) modelos aplicados a ambos. Na comparação com os dados sintéticos o modelo SARIMA tem melhor rendimento com as métricas MSE e TU seguido de SVM e kNN-TSPI. Os testes com dados reais o SARIMA manteve sua superioridade com as três métricas MSE, TU, POCID. Os modelos baseados em Exponential Smoothing e Moving Average obtiveram o pior resultado e ainda o SVM e kNN-TSPI mantém bons resultados, mas abaixo do SARIMA.

Finalmente, realizaram uma última comparação com dados sintéticos, reais e com uma métrica MCPM mostram que os algoritmos neste ordem SARIMA, MLP, SVM e kNN-TSPI de destacam. Dentro do grupo de algoritmos estatísticos o SARIMA é o único método que destaca no grupo dos métodos estatísticos e mais promissório.

2. O estudo apresentado por Adhikari e Agrawal (2014), explora previsão do futuro do preço, modelando o comportamento linear e não linear das séries temporais.

   Os autores demonstram que as séries temporais financeiras têm comportamento linear e não linear. Devido a isso, propõem um modelo híbrido para prever as séries, que combina um modelo linear *Random Walk* (RW) com modelos não lineares *Artificial Neural Network*(ANN) – especificamente uma *Feedforward Artificial Neural Network* (FANN) – e uma rede recorrente *Elman Artificial Neural Net* (EANN). Associando dois modelos, conseguem uma previsão mais estável que os mesmos modelos aplicados isoladamente.

Esse estudo não se identificou um método de pesquisa, mas declarou os passos em termo geral. Primeiro, modela as séries e separa: $Y = X + Z$ onde $X = \{x_1, x_2 \dots x_n\}^T$ é o componente linear para ser usado com o RW, e $Z = \{z_1, z_2 \dots z_n\}^T$ é o componente não-linear. Segundo, tem outro vetor para a predição $\hat{X} = \{\hat{x}_1, \hat{x}_2 \dots \hat{x}_n\}^T$ e realiza a operação de diferença $E = Y - \hat{X}$ para obter os residuais e ser usado com o FANN e o EANN.

Para avaliar a proposta, os autores compararam resultados em quatro bases de dados de diferentes mercados, que foram treinadas com o FANN e o EANN com 2000 épocas. Utilizou-se as medidas de erro MAE, MSE, SMAPE comparando o erro estatístico entre y_t e $\hat{y}_t$ (o preço real e o preço previsto). Compararam quatro modelos – RW, FANN, EANN e híbrido – conseguindo um erro de 0.237, 0.256, 0.290 e 0.247 para o híbrido, EANN, FANN e RW, respectivamente. Para concluir, nesse trabalho, os autores mostraram a complexidade das séries temporais devido à sua natureza de movimentos irregulares que torna difícil de prever valores futuros. Então, o modelo híbrido proposto melhora significativamente a tarefa da previsão em relação a utilizar os modelos isoladamente.

6.2.4 Abordagens Mistas

Os trabalhos anteriores estão baseados em métodos estatísticos e métodos de aprendizado de máquina compartilham um fator em comum, esses abordagens utilizam só o histórico da série temporal para encontrar padrões uteis para a previsão, ou só utilizam o conhecimento externo extraindo informação útil para a previsão. Poucos são os trabalhos que utilizam esse conhecimento externo resultados do processo de mineração de textos para ser utilizado num modelo de previsão e conseguir de alguma forma melhorar a precisão da previsão.

Na literatura se encontrou alguns trabalhos que misturam o abordagens baseados em métodos de aprendizado de máquina e mineração de textos, não se encontrou métodos estatísticos que considerem o conhecimento externo em forma de notícias. Ou seja, realizam uma análise textual e fazem uma análise do passado das séries temporais.

1. No trabalho de Deng et al. (2019), existe a problemática da pouca representatividade de informação semântica na previsão do preço em séries temporais. Os modelos atuais só consideram uma representação numérica baseada no histórico do preço e utilizam modelos de aprendizado profundo como a RNN e LSTM para aprender características de dados não lineares em quanto ignoram a informação semântica no entorno delas. Esta informação semântica está expressada, por exemplo, no seguinte caso. Considere duas empresas do setor Tecnologia e sub industria "Internet software & Service" e seus *head quarters* estão localizados em Califórnia (como Apple e

Ebay). Se um evento relacionado a elas ocorre, então ambas companhias podem ter flutuações similares dos preços.

Os autores, apresentam uma abordagem baseada em fluxo de ontologias semânticas e o aprendizado de características semânticas presente nos textos para embutir em um modelo de rede neural chamado *STBNet*. Este modelo está baseado em mecanismos de atenção e similaridade de semânticas. Esses vetores de atenção e vetores de vinculação são utilizados em uma rede LSTM e CNN.

Essa informação é parte do modelo da rede profunda onde o alvo é aprender uma função que consiga prever o preço $p = F(s, e, \alpha)$ baseado em estos vetores.

A metodologia adotada foi: a) embutir o conhecimento de fundo de um fluxo de ontologia em vetores de vinculação, b) enriquecer o conhecimento de fundo com texto embutido, c) previsão semântica do preço e uma análise numérica.

- A semântica do fluxo de ontologias contém dois tipos de conhecimento que são os vetores de vinculação e os vetores de semântica de atenção. Os vetores de vinculação $e = \{e_1, e_2, ..., e_m\}$ são valores que atuam nas séries, por exemplo a industria, a cidade, o estado associado com um índice S&P 500. Os vetores de atenção $\alpha = \{\alpha_0, \alpha_0, ..., \alpha_n\}$ são gerados pelas distâncias das entidades de condução denotado por $v = \{v_0, v_1, ..., v_n\}$ e as entidades alvo v_{sp}.
- A função do texto embebido é adicionar um conhecimento extra a rede neural. Primeiro, este passo é composto pela extração semântica no fluxo de texto como o nome das entidades e sua semântica associada em base de dados DBpedia e Freebase. Segundo, o aprendizado no fluxo de texto é feito mediante a a vetorização e extração de características nos textos, este passo é realizado com uma rede CNN. Finalmente o vetor de características aprendidas com a rede convolucional gera um vetor de *bag of entailments*.
- A previsão do preço é realizada como uma rede LSTM para o aprendizado de sequências. A entrada do modelo é composta por o vetor de atenção e o vetor de vinculação. Primeiro extraem um conjunto de séries representativas multiplicadas pelo vetor de atenção.

  $$\hat{s} = \{\alpha_0 s_0, \alpha_0 s_1, ..., \alpha_n s_k\}$$

  Onde s_k são as séries mais representativas e α_n é o vetor atenção. Segundo, concatena o vetor de vinculação a $\hat{s}$. A entrada para a LSTM é $s' = [\hat{s}, e]$ e o resultado é obtido pela função $p = F(s')$.

Para os experimentos, utilizaram a) uma base de dados do preço de S&P500 de quatro meses, b) foi construído um grafo de conhecimento de informação de cada *stock* como a segurança, o setor, a subindústria. c) foram utilizados dados de texto

extraídos do Twitter relacionados a S&P500. O modelo CNN foi treinado com os algoritmos *Stochastic Gradient Descent* (SGD) com uma optimizador *Momentum Optimizer*. As medidas de avaliação utilizadas foram MAE, MAPE e RMSE.

O modelo STBNet foi comparado com quatro modelos para comparar os resultado com outras abordagens. Como *baseline*, utilizou-se um modelo ARIMA e LSTM, e para uma comparação mais profunda foi usado um SE-LSTM (com um mecanismo de atenção) e TE-LSTM (com vetores de vinculação). Os autores mostram que o modelo proposto STBNet obteve o erro mais baixo, próximos de 2.75, 1.41 e 3.25 (MAE, MAPE, RMSE respectivamente) e que os modelos SE-LSTM e TE-LSTM têm melhor performance que o LSTM e ARIMA. Assim os resultados validam a hipótese que o conhecimento semântico melhora a efetividade da previsão das séries temporais.

2. Rodrigues et al. (2018) mistura a previsão de séries temporais e a mineração de dados utilizando técnicas de PLN. A problemática enfrentada é o risco que tem as derivados cotizados em bolsa do Agronegócio. Para resolver o problema, incluem informação externa contidas em notícias especificamente do setor agronegócios sem a necessidade de que estejam rotuladas, com o alvo de melhorar a previsão dos da série temporal.

   A metodologia proposta neste trabalho foi a identificação automática de períodos de interesse em altas e quedas da série temporal mediante um conceito chamado *Percentual Important Point* (PIP). As notícias publicadas nesses períodos são consideradas de grande influencia para apoiar a previsão. Além disso consideram os PIP's recorrentes ou seja mais de duas notícias publicadas com certa regularidade em períodos de altas ou quedas são automaticamente selecionadas para serem incorporadas na previsão. Para a previsão da série temporal histórica utilizam o algoritmo KNN com a distância *Dynamic Time Warping* (DTW) publicado em outro trabalho(Marcacini; Carnevali; Domingos, 2016) das k-subséries mais próximas.

3. Os autores em Akita et al. (2016), mencionam que existe informação numérica e textual que os investidores lidam todos os dias como *charts*, volumes de preço, notícias nos jornais. É muito difícil analisar todos esses dados para prever os preços. Os modelos presentes utilizam séries temporais sem levar em conta a informação textual. Devido a isso, o trabalho deles, definem como hipótese que a utilização de informação textual e numérica pode influenciar positivamente no modelo e captura relações mais complexas entre o texto e o preço das ações. O modelo proposto é baseado em *Paragraph Vector* (PV) para a representação do texto distribuída e uma representação numérica da série temporal para ser usadas como entrada para a rede LSTM para fazer a previsão do preço como informação textual e numérica.

O modelo proposto contém dois vetores, um vetor que é a representação textual de artigo e um vetor da representação numérica. O vetor p_t expressa a concatenação dos pesos de várias companhias. Esse vetor VP foi implementado com duas técnicas a *Distributed Memory Model of Paragraph Vectors* (PV-DM) e *Distributed Bag of Words of Paragraph Vector* (PV-DBOW).

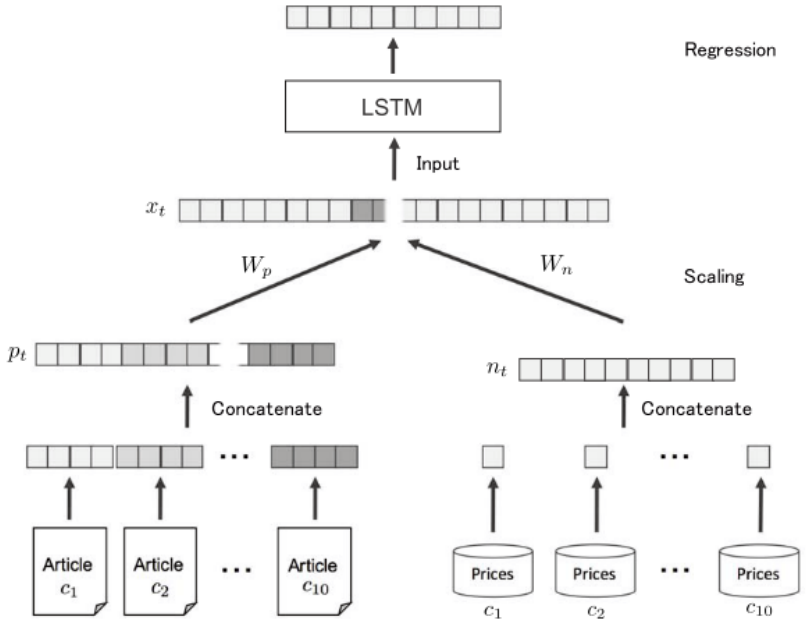
Cada articulo é representado por um vetor fixo que usam algum modelo de PV para representar um artigo de uma companhia, por exemplo, de acordo com aos autores, se temos varias empresas $\{c_1, c_2, ..., c_{10}\}$, então se c_2 não tem artigo publicados e dois artigos publicados são de c_{10}:

$$P_t = \{a^t_{c1}, 0, ..., \frac{a^t_{c10,1} + a^t_{c10,2}}{2}\}$$

Onde $a^t_{c_n}$ é uma representação distribuída da companhia c_n, e t é o tempo.

A representação numérica, foi realizada com uma normalização do preço mínimo, máximo de cada empresa no vetor N_t. Finalmente o modelo para a LSTM é construído com a concatenação dos vetores P_t e N_t com a ajuda de uma rede neural para coincidir fazer um *scaling* do vetor que não tem o mesmo tamanho. Uma representação do modelo proposto na Figura 26.

Figura 26 – Modelo de Representação de Palavras e Série temporal com a operação de concatenação.



Fonte: Akita et al. (2016).

Os resultados foram divididos em dois parte. A primeira parte foram realizados a comparação da efetividade do Paragraph Vector PV+Num com dois modelos:

modelo só de Numérico Num, Bag of Word BoW+Numerico. O PV+Num mostro melhor rentabilidade que os outros modelos.

A segunda parte foi avaliar o *performance* da LSTM com outros modelos *baseline* como: MLP, SVM, RNN. Os modelos que conseguiram retornos altos em todas as provas feitas com séries de diferentes industrias foram o LSTM e a RNN, conclui a importância da escolha do modelo LSTM junto com um modelo de representação dos textos melhora a acurácia e os rendimentos econômicos.

6.3 Discussões dos Trabalhos Anteriores

Cada trabalho apresentado resolve a tarefa da previsão das séries temporais com uma técnica diferente e reporta a taxa de precisão. Este problema radica em encontrar a melhor função hipótese $y = F(x)$ que calcule o próximo valor Z_{t+h} da série $Z = (z_1, z_2, z_3...z_t)$. A justificativa da escolha desses artigos está baseada nas diferentes abordagens presentes no estado do arte..

Ultimamente, a academia usa as técnicas de aprendizado profundo, em lugar das técnicas estatísticas, por sua alta capacidade capturar as relações não lineares em sequências, já que os modelos ARIMA, SARIMA não permitem modelar padrões não regulares inerentes nas séries temporais e sobretudo por precisar de um conhecimento a priori da distribuição dos dados como menciona o estudo em Parmezan, Souza e Batista (2019). Tradicionalmente, as redes recorrentes RNN sofrem do problema de *vanish gradients*, já que é difícil capturar dependências longas. Mas, isso é superado pela existência de outros modelos como a LSTM, Encoders, Mecanismos baseados em Atenção. Também se encontrou que muitos artigos utilizam outros enfoques como ANN, SVM, kNN, MLP e técnicas de aprendizado profundo como Redes Convolucionais.

a) O estudo comparativo dos autores em Parmezan, Souza e Batista (2019), mostram que a abordagem estatística, especificamente o SARIMA, é a melhor técnica para fazer previsão, desde que haja um conhecimento *a priori* e estável da média e desvio padrão. Por outro lado os métodos baseados em aprendizado de máquina mostram resultados similares que não precisam deste conhecimento a priori e somente se precisa saber os parâmetros para configurar as redes ou modelos.

b) O trabalho dos autores Parmezan e Batista (2015) está baseado na busca local de sequências mais parecidas a uma sequência Q. Os autores modificam o k-NN para conseguir melhores resultados. Uma variável muita questionada é o tamanho da sequência $l = len(Q)$, o tamanho de l tem que coincidir com um valor sazonal das séries temporais mas não sempre as séries temporais têm comportamentos sazonais.

c) No trabalho de Kim et al. (2016), os autores expressam a relação de causalidade de Granger entre duas variáveis. O teste de Granger, proposto por Granger (1969), é

um preditor linear que só testa a causalidade linear, embora a relação entre duas variáveis econômicas seja muito complexa para ser resumida em uma relação linear simples. A necessidade da procura de novos testes que falhem ao aplicar o *t-test* de causalidade, já que não é necessariamente verdadeiro o teste de Granger.

d) No trabalho de Bao, Yue e Rao (2017) destaca-se o uso de *Wavelets* para pré-processamento da data e o Autoencoder para aprender as características. O uso do *Wavelets* pode reduzir o *overfitting* da fase do treino, coisa que não se aplicou em outros trabalhos, minimizando o *Root Mean Squared Error* (RMSE)

e) Os autores do artigo Deng et al. (2019) utilizam um mecanismo baseado em atenção na semântica nos textos associada aos valores das séries temporais e a semântica presente nos textos para embutir essa informação no modelo. Esta proposta comparada com Bao, Yue e Rao (2017), onde utiliza um encoder-decoder junto com um filtro *Waveletes* para tratamento da não linearidade, substitui o uso com uma rede baseada na atenção das séries mais significativas. A crítica a este trabalho seria por não comparar os resultados obtidos com uma avaliação de retorno, já que somente mostra resultados baseados em métricas de erro conhecidas.

Capítulo 7

Metodologia da Proposta

Neste capítulo, apresentamos nossa proposta do trabalho. Para avaliar a hipótese de pesquisa (1.4), nosso trabalho utiliza a metodologia "experimental" para a geração dos modelos até a sua avaliação num ambiente simulado.

7.1 Pesquisa Experimental

Esta pesquisa, assim como toda pesquisa científica, se inicia com a definição de um problema que ainda não se sabe se tem uma solução. O passo seguinte consiste em oferecer uma solução possível, mediante uma proposição, ou seja, uma expressão verbal suscetível de ser declarada verdadeira ou falsa Antonio (2008).

Segundo Lakatos e Marconi (2003), o objetivo principal de investigações experimentais (ou empíricas) é o teste de hipóteses, que dizem respeito a relações de tipo causa-efeito de duas variáveis. Todos os estudos desse tipo utilizam projetos experimentais que incluem os seguintes fatores: grupos de controle (além do experimental), seleção da amostra por técnica probabilística e manipulação das variáveis independentes com a finalidade de controlar ao máximo os fatores pertinentes.

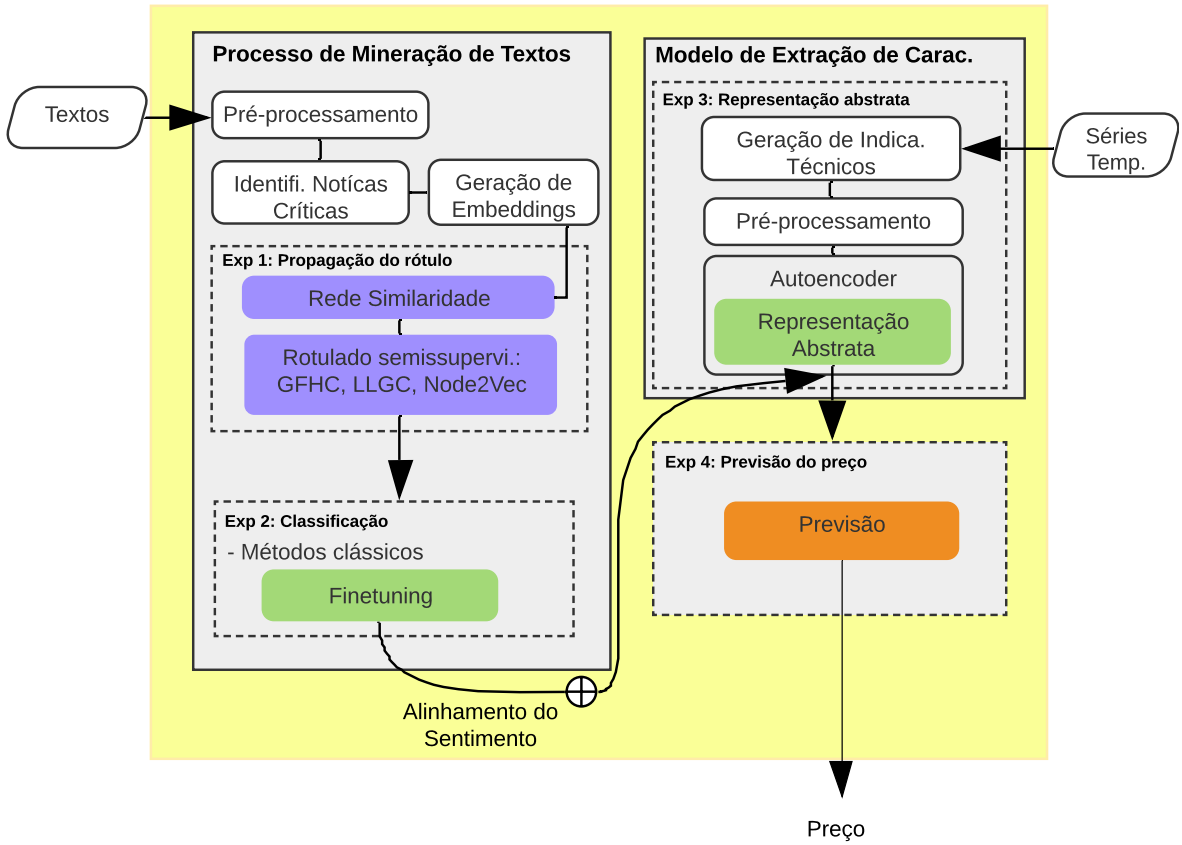
Para Lazar, Feng e Hochheiser (2017), a diferença entre a pesquisa experimental e os outros dois tipos de investigação é que a pesquisa experimental permite a identificação de relações causais. Simplificando, ela pode dizer como algo acontece e, em alguns casos, por que isso acontece. O alvo da pesquisa experimental é identificar a verdadeira causa de um fenômeno, permitindo aos pesquisadores manipular o jeito como pesquisamos e alcançamos os resultados.

7.2 Visão Geral

Nesta seção, é apresentada uma abordagem híbrida que permite incorporar informações externas no modelo de previsão de séries temporais com a finalidade de melhorar a acurácia da previsão. Nossa proposta pode-se dividir três partes. 1) realizamos o processo de mineração de textos, quem extrai o conhecimento externo das notícias relacionadas ao cripto ativo *Bitcoin*; este passo esta formado por varias sub etapas; 2) à extração de uma representação compacta das características utilizando um *autoencoder*, que de modo não supervisionado aprende as melhores representações dos indicadores técnicos da cripto moeda; 3) a previsão do preço.

A Figura 27, mostra um panorama global da proposta. Esta proposta visa fundir duas abordagens utilizadas pelos especialistas deste domínio: a Análise Fundamentalista e a Análise Técnica para um caso de estudo utilizando o ativo digital *Bitcoin*.

Figura 27 – Visão geral da proposta. A metodologia está formada por três etapas e cada uma com seus experimentos relacionados. Na esquerda mostra-se o processo de mineração de textos, a direita o modelo que processa as séries temporais. No processo de mineração de texto contém os experimentos 1 e 2 associados. Os experimentos 3 e 4 estão associados a extração de caraterísticas e a previsão do preço.



Fonte: Elaborada pelo autor.

A parte esquerda da figura, visa modelar a Análise Fundamentalista, onde o alvo é

analisar o sentimento ou a opinião das notícias. Esta etapa do processo de mineração de textos contêm vários passos para extrair o sentimento presente nas notícias financeiras do ativo digital.

Após extrair esse conhecimento das notícias, ele é adicionado ou alinhado com as séries temporais, para seu posterior uso. À figura da direita, é mostrado o modelo de representações de características compactas, que aprende com um *Autoencoder* os indicadores mais importantes da Análise Técnica.

Finalmente, após extrair a opinião mediante o processo de mineração de textos e a extração de características compactas com o *autoencoder*, a previsão do preço é feita com a informação do sentimento presente nos textos e as características mais relevantes.

7.3 Processo de Mineração de Textos

O objetivo deste trabalho, mencionado na Seção 1.4, é demonstrar se a incorporação do conteúdo de um texto, especificamente uma notícia, tem algum tipo influência positiva ou negativa no preço, ou se a incorporação do conhecimento externo não melhora a previsão do preço.

O processo de mineração de textos é composto por diferentes etapas, desde a obtenção dos dados até sua avaliação final. Nas próximas seções, falaremos sobre cada etapa mais detalhadamente. A lista a seguir sumariza o desenvolvimento deste capítulo.

a) Construção do Corpus, Pré-processamento;

b) Representação Textual com *Embeddings*;

c) Identificação de Notícias Críticas;

d) Rotulado Semissupervisionado;

e) Multi classificação com *Fine Tuning*;

7.3.1 Construção do Corpus

Para a construção do corpus nos guiamos algumas das 7 questões da construção do corpus explicados na Seção 5.2.1, e adaptamos a nossas necessidades. A estratégia utilizada para rotular os textos é baseada na propagação do rótulo a partir de um conjunto pequeno de textos rotulados para textos não rotulados, este procedimento será explicado na Seção 7.3.4. Esta estratégia é usada devido a problemática da existência de milhares de notícias sem rótulo.

1. **Coleta dos dados**: Uma parte importante neste passo é a escolha do ativo. Atualmente existem vários ativos no mercado. Entre eles, as características procuradas foram: a maturidade, liquidez e volatilidade. O ativo escolhido para esta pesquisa foi o “Bitcoin”, uma vez que ele pertence a um tipo de mercado emergente no

mundo cripto tecnológico e que está em constante desenvolvimento, além de possuir as características mencionadas, ele contém dados públicos e sobretudo é altamente volátil.

Para a coleta das notícias, foram utilizados *scripts* de automatização desenvolvidos em Python e com ajuda de *APIs* (newspaper, feedparser). Esses scripts foram configurados para recuperar informações de sítios com uma confiança alta e, portanto, não são considerados propagadores de *fake news* e contém uma escrita literária que obedece a regras gramaticais da língua Inglesa.

Para a coleta, foi usada o *keyword bitcoin* e os sítios utilizados para realizar o *crawling* das notícias estão listados na Tabela 1.

Tabela 1 – Sítios web utilizados para o *Crawling* de Notícias.

Sítios	Enlace
Cointelegraph	cointelegraph.com/rss/tag/bitcoin
Coindesk	coindesk.com
Newsbtc	newsbtc.com/feed
CCN	ccn.com/tag/bitcoin
DailyFx	dailyfx.com
Fxstreet	fxstreet.com
AmbCrypto	ambcrypto.com
CryptoDaily	cryptodaily.co.uk
Utoday	u.today/latest-bitcoin-btc-news

Um problema detectado após a coleta de dados, foi o fato que existem milhares de notícias que não aportam algum tipo interesse ao fenômeno estudado, mas estão associados ao *keyword* "bitcoin" de alguma maneira. Para resolver este problema, utilizamos filtros para excluir notícias que contêm *keywords* de uma *blacklist* e outra lista que incluem notícias de interesse sobre todo as notícias que impulsionam o mercado para uma queda ou alta. Por exemplo decisões relacionadas a alguns conceitos como *Security Exchange Comission* (SEC), *Exchange Trade Funds* (ETF's), e *Chicago Board Options Exchange* (CBoE), que estão relacionados ao keyword "bitcoin". No caso de notícias excluídas, existe, por exemplo, muita notícia relacionada a análise do preço com o *keyword* "cryptocurrency price analysis" que não influencia na série temporal.

A Tabela 2, mostra uma lista das *keywords* utilizados para o filtro. As notícias com *keywords* excluídas não aportam informação útil. O *dataset* resultante contêm 10.100 notícias de 1 de Novembro do 2018 a 3 Novembro 2020.

2. **Pré-processamento**, Após a construção do corpus e antes de submeter a qualquer tipo de processamento computacional, é necessário preparar esses textos num for-

Tabela 2 – Lista de palavras chaves utilizados para filtrar as Notícias

	Palavras chave
Excluídos	'price analysis', 'cryptocurrency price analysis', 'bitcoin cash', 'cryptocurrency price review', 'cryptocurrency marketcap', 'bitcoin sv', 'ripple', 'bitcoin gold', 'xrp', 'bch', 'tron', 'ethereum', 'eos', 'neo', 'eth'
Incluídos	'bitcoin', 'blockchain', 'bakkt', 'vaneck', 'vaneck bitcoin etf', 'solidx' 'bitcoin futures', 'cryptocurrency', 'sec', 'cboe', 'bitcoin trust', 'cme', 'distributed ledger', 'securities exchange commission', 'halving' 'etf', 'cme group', 'blockchain leader', 'coronavirus'

mato adequado para extrair algum tipo de conhecimento. O texto deve passar por algum tipo de limpeza e filtragem para que possa ser entendido por um algum algoritmo. Além disso, o pré-processamento ajuda a lidar com a alta dimensionalidade dos dados, já que uma pequena coleção de textos pode conter milhares de termos, muitos deles redundantes que tornam a tarefa de extração do conhecimento lenta e pode até prejudicar os resultados finais. Esta etapa foi realizada conforme os passos explicados na Seção 5.3. Nestas etapas se utilizaram vários *scripts* e funções para adaptar o texto. A lista a seguir sumariza as etapas:

- Normalização e substituição: esta etapa visa padronizar os dados removendo caracteres não úteis da língua e havendo a possibilidade de aproximar o máximo possível esses textos as normas de uma língua formal, em nosso caso a língua inglesa. Foram removidos caracteres especias do tipo: '!' , '?', ',', '.', '@', '#', '$', 'http', 'emoticos', 'urls' e números. Para aproximar a uma língua formal se aplicou uma substituição das contração do inglês informal para formal do tipo e algumas outras palavras:

```
"ain't": "is not", "aren't": "are not","can't": "cannot",
"can't've": "cannot have", "'cause": "because",
"could've": "could have", "couldn't": "could not",
"couldn't've": "could not have","didn't": "did not",
"doesn't": "does not", "don't": "do not",
"usd/btc's": "btcusd is", "btc/usd's": "btcusd is",
"btc": "bitcoin", "xbt": "bitcoin"
```

  A lista acima mostra o dicionário de algumas palavras substituídas utilizados, por exemplo para a frase "could't" foi substituída por sua forma "could not" assim como "usd/btc's" para "btcusd is".

- Foram removidos as "stopwords", que não aportam valor semântico.
- Já normalizados e substituídas as palavras, os textos foram tokenizados.

É muito importante mencionar que dependendo da técnica usada para representar os textos (que será explicado na Seção 7.3.2), o pré-processamento muda. Por exemplo, para representações que usam *Embeddings* contextuais não se removeram as "stopwords", nem caracteres do tipo: ',', '.' mantendo estas sentenças originais, já que ajuda entender melhor o contexto (QIAO et al., 2019). Os números inteiros e flotantes tampouco foram removidos de acordo a (WALLACE et al., 2019). É recomendável fazer um pré-processamento fraco quando se usa o BERT.

3. **Anotação do Corpus**: a tarefa de anotação (ou "*tagging*") é o processo custoso e pode consumir muitos recursos de tempo e econômicos. Parte do desafio desta pesquisa é rotular só um conjunto inicial pequeno de dados para espalhar o rótulo a dados sem rótulo. Este processo será explicado melhor nas próximas seções.

4. **Avaliação do corpus**, é importante avaliar o grau de concordância entre o avaliador e o corpus, para tal coisa precisamos de alguma métrica. Normalmente a literatura recomenda utilizar o coeficiente *Kappa* e o coeficiente de correlação *Pearson*, já que eles são normalmente utilizados para garantir a confiabilidade do corpus anotado por vários avaliadores.

   Para nosso casso, esta questão recomendada na Seção 5.2.1, será ignorada já que nosso corpus é anotado de forma semissupervisionada, neste caso utilizaremos outra métrica que será explicada na Seção 8.1.

7.3.2 Modelos de Representação Textual

Os textos estruturados e concisos da etapa anterior a construção do corpus ainda estão em língua natural que não é entendível a uma máquina. O objetivo de esta etapa é transformar esses textos estruturados em representações numéricas.

Para a representação do texto, o estado da arte costuma a usar diferentes técnicas: *Bag of Words* (BoW), *Term Frequency-Inverse Document Frequency* (TF-IDF) e *Word Embeddings* como modelos de representação de espaço vetorial ou características numéricas. Estas reapresentações foram explicadas na Seção 5.4.

O modelo mais utilizado para a representação dos textos é o modelo espaço-vetorial, no qual cada texto é representado por um vetor de m dimensões e cada dimensão é um atributo da coleção (CORRÊA et al., 2012). Cada atributo tem um peso indicando sua relevância para um determinado texto.

Para representar esse modelo espaço vetorial, pode-se estruturar em uma *bag of words*, onde os atributos são todas as palavras de um documento, assim as dimensões desse espaço vetorial é igual ao número de todas as palavras de todos os documentos (FELDMAN; SANGER, 2006). O modelo *bag of words* é um tabela de entradas de documento-termo, onde cada entrada na tabela é valor ou peso, expressado por diferente técnicas listadas:

a) A forma mais simples é a binária, o peso é 1 se o termo está presente no documento e 0 em caso que o termo está ausente.

b) Outra técnica mais complexa pode ser expressado por frequências de termo em um documento, em uma categoria ou em toda a coleção de documentos, esta técnica é conhecida como *TF-IDF*. Esta técnica serve para construir estatísticas de frequência do corpus.

c) Os textos também podem ser representados por seus vetores *Word Embeddings* que guarda a relação semântica das palavras.

Em nosso trabalho, propõe utilizar representações contextuais *Bert Embeddings*, porém foram testadas outros tipos de representações como: *Word2Vec*, onde utilizamos o corpus coletado para produzir *Embeddings* distribuídas. O *TF-IDF*, esta produz resultados relativamente rápidos, mas contém uma alta dimensionalidade (se não for especificada) e não respeita a sintaxes e semântica das palavras. Não usamos *bag of words* devido a sua incapacidade de lidar com a alta dimensionalidade já que os métodos de extração de padrões são prejudicados pela alta dimensionalidade, a pobreza semântica ignorando a ordem das palavras (CORRÊA et al., 2012).

Para mais detalhe, veja as Tabelas 6, 8 na Seção de experimentos.

7.3.3 Identificação de Notícias Críticas

O alvo desta etapa é construir o conjunto inicial de textos rotulados automaticamente, que serão utilizados na próxima etapa de propagação do rótulo. Quando falamos automaticamente não quer dizer que não tem a intervenção de um humano que valide os textos rotulados. Esta validação é importante para os algoritmos semissupervisionados.

Para identificar as notícias mais relevantes para uma série temporal, se utilizou uma estratégia que consiste em: a) utilizar a série temporal como suporte para identificar os intervalos datas que teve volatilidade, para isso se utilizou o cálculo a diferença entre *open-to-close* baseado na idéia do (Lien Minh et al., 2018) com algumas modificações a nossos requerimentos, adicionando o indicador técnico *Relative Strenght Index*(RSI), que serve de apoio para identificar só períodos com tendência:

$$\begin{aligned} r_t &= O_{t+f} - C_t \\ rsi_t &= RSI_{t+f} \\ \text{Datas Volateies} &= \begin{cases} uptrend, & \text{se} \quad r_t > 0, rsi > rsi.high \\ downtrend, & \text{se} \quad r_t < 0, rsi \leq rsi.low \end{cases} \end{aligned} \tag{52}$$

Para o cálculo da diferença r_t se utiliza O_{t+f} que é o preço de abertura da série temporal no tempo t e f é f períodos depois ($f \in [1, 2, 5, 7, 10]$). C_t é o preço de fechamento. O valor de rsi_t é calculado com um indicador técnico *Relative Strengh Index* (RSI), que

avalia a velocidade e a tendência do movimento do preço. O RSI está configurado com os valores máximos e mínimos *rsi.high* e *rsi.low* (75, 25 respectivamente).

A Figura 28, mostra o processo da estratégia proposta.

Figura 28 – Exemplo da estratégia: Uso da série temporal para criar o conjunto de notícias inicias.

Fonte: Elaborada pelo autor com o software *TradingView*.

Após de identificar as datas tendências de alta ou queda (up trend, down trend), b) se seleciona apenas as notícias publicadas nos períodos de tendência. Esse conjunto de notícias pode ser considerado o conjunto inicial de notícias pré-candidatas críticas, mas nem toda notícia encontradas nestas tendências tem influência na série temporal. Para garantir que alguma notícia teve grande impacto, filtramos as notícias que pelo menos possuem duas notícias publicadas dentro de suas k-notícias mais similares semânticamente, esta ideia é baseada em (RODRIGUES et al., 2018). Para esta parte propomos utilizar *Bert Embeddings* como representações de texto. Para calcular a similaridade entre estas representações usamos a distância do Cosseno definida pela Equação 53.

Esta estratégia visa eliminar outras notícias publicadas em períodos de interesse, mas que não têm relação semântica com as k notícias de interesse. Nossa hipótese é que notícias publicadas nos pontos de alta ou queda, com frequência e com similaridade muito alta, possam ser úteis tanto para espalhar o rótulo como ajuda a previsão do preço.

Desta forma proporcionamos uma ferramenta ao esperto do domínio que automatiza o rotulado de notícias para que evite selecionar manualmente, lendo milhares de textos e olhando a serie temporal para rotular cada texto como positivo, negativo ou neutro. Porém como mencionamos no inicio, é preciso que um humano valide os textos rotulados automaticamente para seu posterior uso.

7.3.4 Anotação do Corpus Semissupervisionada

O objetivo nesta etapa é construir um corpus anotado maior a partir de um conjunto de textos rotulados, agrupando as notícias positivas, negativas e neutras com alguma técnica de agrupamento semissupervisionada.

Após criar o conjunto inicial de notícias críticas e validadas por um supervisor na etapa anterior, representamos os documentos num espaço vetorial, para nosso caso geramos sua *embedding*, pode se aplicar alguma técnica de agrupamento de documentos que nos permita organizar os grupos de acordo a uma medida de similaridade. Não entanto em técnicas de agrupamento não há rotulo pré-definido e são conhecidas como técnicas de aprendizado não supervisionado ao contrário do aprendizado semissupervisionado ou assistido por um humano com conhecimento do domínio.

Para nossa estratégia de agrupamento será utilizado o algoritmo semissupervisionado *Label Propagation* explicado na Seção 5.5. Os passos gerais para o processo de propagação de rótulo são:

1. Ler a base de dados e gerar as representações dos textos.
2. Gerar um grafo não direcionado, calculando as distancias entre os nós.
3. Gerar a matriz de adjacências, criando as relações entre nós de acordo a sua similaridade. Cada nó contém sua representação no espaço vetorial. Para unir as arestas entre nós, utilizamos a distância da similaridade cosseno entre estas representações. A distância mínima utilizada como critério de similaridade será 0.85.
4. Calcular o peso das arestas.
5. Após construído o grafo, aplicar o Algoritmo LP, usando o conjunto pequeno de notícias rotuladas e validadas por um supervisor no passo explicado na Seção 7.3.3.
6. Gerar as classes de saída.

Como já se mencionou, o grafo está formado por nós que representam um vetor de características de um documento. Os textos mais próximos representam documentos similares. Foi utilizado a similaridade do Cosseno definida pela Equação 53 (FELDMAN; SANGER, 2006; TAN; STEINBACH; KUMAR, 2006) como o ângulo cosseno formado entre os dois vetores de n-dimensões, considere dois documentos representados por: $x_i = (x_{i,1}, x_{i,2}, ...x_{i,n})$ e $x_j = (x_{j,1}, x_{j,2}, ...x_{j,n})$.

$$sim(x_i, x_j) = \frac{x_i.x_j}{||x_i||||x_j||} \tag{53}$$

Onde $||x_i||$ é a norma Euclidiana do vetor x_i definida como: $\sqrt{(x_{i,1}, x_{i,2}, ...x_{i,n})^2}$ similarmente se calcula para $||x_j||$. Esta medida calcula o cosseno do ângulo entre os vetores x_i e x_j. Os resultados das notícias anotadas serão mostrados na Seção 8.1.

7.3.5 Multi Classificação: Fine Tuning

Esta proposta propõe utilizar o modelo pré-treinado do *BERT*, explicado na seção 5.6.3 fazendo fine-tunning para a tarefa específica. O *BERT* é um dos modelos de linguagem mais poderosos de contexto e representações de palavras, baseado na arquitetura de *Transformer* e utiliza mecanismos de atenção.

Uma das vantagens do *BERT* é que ele é capaz de ser utilizado de duas formas:

a) Pode ser utilizado para gerar *word embeddings* dada uma sentencia, estas *embeddings* serve como entrada para outra tarefa.

b) Pode transferir o conhecimento que têm no seu modelo pre-treinado, como resultado é capaz de fazer *fine-tunning* para e especializa-lo na última camada para ser utilizado como um classificador.

Esta última característica permite reutilizar o modelo pré-treinado com um corpus gigante, para una tarefa específica num corpus menor, devido que ele contém poucos textos financeiros específicos de nosso domínio. Assim, podemos reutilizar as últimas camadas e os pesos delas para tunear nosso modelo para multi-classificação usando uma função *softmax* para diferentes classes.

7.4 Modelo de Extração de Características das Séries Temporais

Até o presente momento nas seções anteriores, apresentamos o processo de mineração de dados textuais, que visa simular a Análise Fundamentalista. O objetivo geral nesta etapa é extrair algum tipo de conhecimento de uma lista características ou Indicadores Técnicos para ser específicos.

A ideia geral é a extração de uma representação compacta, que de modo não supervisado aprende os atributos mais representativos dos Indicadores Técnicos associados a uma série temporal multivariada.

Para cumprir isso, o modelo mostrado na Figura 27 (direita), está dividido nos seguintes passos: 1) a coleta da série temporal, 2) a geração de indicadores técnicos e pré-processamento e 3) a geração de uma camada compacta de *features* de menor tamanho com o *Autoencoder*, esta nova camada esta codificada com uma função *encoder*. Este novo "input" serve como entrada para qualquer modelo de previsão, que será explicado na seção 7.6.

7.4.1 Coleta de dados e Geração de Indicadores

A coleta de dados é um processo trabalhoso. Neste trabalho, utilizamos dados históricos do ativo Bitcoin. Para a coleta, foi utilizado *software open source gekko*[1]. Os dados coletados estão no formato OHCL (*Open, High, Close, Low*).

Para a geração dos indicadores técnicos foram utilizadas as bibliotecas *Talib* [2] e *Tulip*[3] que facilitaram sua construção do "dataset". Foram gerados os indicadores técnicos mais importantes, mencionados nas seções anteriores (2.3).

O *dataset* contém as seguintes características divididas em grupos: *Open, High, Close, Low* (que são os preços diários do ativo), mais o volume comercializado, que formam o grupo relacionado ao preço. O outro grupo é formado pelos indicadores técnicos utilizados na Análise Técnica, como discutido na Seção 2.3. A Tabela 3 mostra os indicadores mais comuns utilizados em nossa proposta.

Tabela 3 – Descrição das características utilizadas para o *Autoencoder*.

Característica	Descrição
Open /Close price	Preço diário de abertura e fechamento.
High /Low price	Preço diário máximo e mínimo.
Volume	Volume de transações.
Indicadores Técnicos	
MACD	Moving Average Convergence Divergence.
RSI	Relative Strength Index.
Stochastic RSI	Stochastic Relative Strange Index.
EMA20	Exponencial Moving Average 20 períodos .
MAE5/MAE10/MAE200	Moving Average 5, 10, 200 períodos.
WMA20	Weighted Moving Average para 20 períodos.
ROC	Price Rate of Change.
SMI	Stochastic Momentum Index.
WDAD	William Variabel Acumulation Distribution.
CCI	Commodity Channel Index.
BBANDS	Bolling Bands.

Os dados foram coletados de servidores públicos como *Bitfinex* e *Yahoo Finance*, que possuem dados históricos com janelas desde 1 minuto até janelas de 1 dia em formato OHCL. Posteriormente estes dados foram discretizados a observações de janelas de 4 horas por dia, utilizando o software *gekko*.

[1] Gekko é um projeto de *open source* para *trading*
[2] https://ta-lib.org/
[3] https://tulipindicators.org/ são bilbiotecas para a geração de indicadores técnicos

7.4.2 Pré-processamento

Esta etapa deve ocorrer antes de se iniciar qualquer análise computacional com algum tipo de algoritmo. Para evitar que nosso algoritmo seja enviesado para as variáveis com maior ordem de grandeza, é recomendável transformar todas as variáveis para a mesma ordem de grandeza. Ao mesmo tempo, se realizou a procura de dados faltantes.

Realizou-se uma análise do dataset das séries temporais e se encontrou algumas características com valores vazios , foram necessário substituir com a moda de uma coluna os valores ausentes *NaN* que se encontraram em lugar de eliminar-los do dataset. A justificativa do uso da moda foi que ela é mais representativa de um conjunto de dados em lugar da média, porque a média é mais sensível aos valores da amostra, ela é mais adequada a situações onde os valores são distribuídos de forma uniforme.

Após realizada essa substituição, se aplicaram duas técnicas para normalizar e padronizar. A normalização é definida pela Equação 54 conhecida como *min-max* para ajustar os atributos a respeito de cada atributo x como o valor máximo $max(x)$ e o valor mínimo $min(x)$ dentro do intervalo $0, 1$ ou $-1, 1$. O novo valor x' corresponde ao valor normalizado.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{54}$$

Já padronização está definida pela Equação 55 que tem como objetivo padronizar as variáveis em uma média igual a 0 e desvio do padrão igual a 1. Normalmente é conhecida como a fórmula *z-score*

$$z = \frac{x - \mu}{\sigma} \tag{55}$$

Nossa escolha dependeu dos resultados obtidos. Ao avaliar as duas possibilidades, notamos que a padronização *min-max* foi mais adequada no nosso domínio de aplicação. Essa e outras decisões são baseadas nos resultados experimentais apresentados no Capítulo 8.

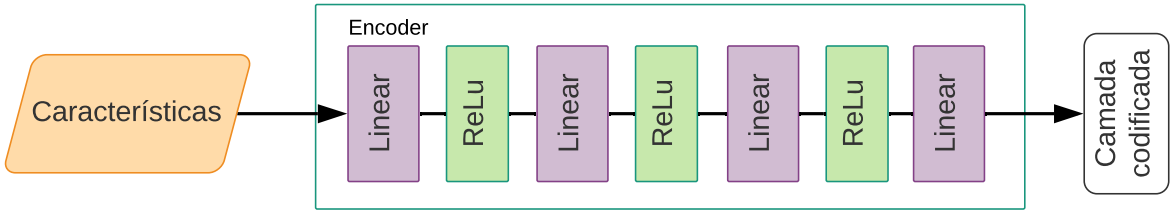
7.4.3 Extração de características

Após o pré-processamento dos dados de entrada a próxima etapa é a extração de uma representação compacta de menor tamanho. O objetivo de esta etapa é encontrar as características mais representativas de uma lista de *feautures* mostradas na Tabela 3, estas novas características são extraídas de forma não supervisionada mediante um *Autoencoder*. Estas novas características serão posteriormente utilizado como "input" de seguinte etapa, a previsão.

A arquitetura do *autoencoder* é formada por uma série de funções *ReLu* e camadas Lineares que permitem aprender as relações de forma não supervisionada, ao mesmo tempo reduzem a um tamanho compacto de novas características. As camadas lineares

contêm como entrada e saída neurônios: L(24, 18), L(18, 12), L(12, 4), à saída final contém 4 características abstratas que descrevem todos os indicadores técnicos. A Figura 29, mostra esta arquitetura.

Figura 29 – Arquitetura interna do Auto-Encoder.



Fonte: Elaborada pelo autor.

Para medir a qualidade de reconstrução dos dados originais, o *Autoencoder* utiliza também as métricas de regressão explicadas na Seção 4.5.1. Foram utilizadas: R^2,*MAE, MSE e RMSE*. Para ver os resultados desta etapa, por favor dirija-se a Seção 8.3, onde serão apresentados os experimentos detalhadamente. Utilizou-se estas métricas para a reconstrução da camada oculta com a função *decoder*, que internamente é a mesma arquitetura do *encoder* com as funções invertidas: L(4, 12), L(12, 18), L(18, 24), com o alvo de reconstruir o *input* inicial, as 24 características originais.

7.5 Enriquecendo as Séries Temporais: Alinhamento de Notícias

Para enriquecer a série temporal com dados textuais, é preciso de uma estratégia de fusão. A estratégia proposta baseia-se no alinhamento ou incorporação do sentimento produzido pela etapa de mineração de textos com as séries temporais.

Temos desenvolvido duas estratégias para a incorporação de notícias. Primeiro, sabendo que o mercado *Bitcoin* opera 24/7 dias, durante todo o ano, diferentemente do mercado tradicional (com dias sem operar como finais de semana e horários específicos). Definimos uma janela de transações diária que inicia-se às 21 horas e se encerra às 21 horas do dia seguinte. As notícias serão alinhadas a cada transação de acordo a data de publicação de cada notícia. Uma transação pode conter várias observações, para nosso caso definimos 6 observações de 4 horas.

A Equação 56, mostra o cálculo da média ponderada das notícias que aconteceram para a transação i durante um dia. As classes definidas como mencionamos anteriormente, foram $k \in \{\text{non trend}(0), \text{up trend}(+), \text{down trend}(\text{-})\ \}$.

$$Tr_i = \left(\frac{\sum N^k}{N_T} \times \alpha[\mathbf{k}] \right) \tag{56}$$

$$Tr_i = argmax_{k \in \{+,-,0\}} \left(\frac{\sum N^k}{N_T} \times \alpha[\mathbf{k}] \right) \tag{57}$$

O Tr_i contém a frequência de cada classe. Esta é calculada contando o número total de notícias para cada classe, por exemplo: contar o número total de notícias *up trend* dividido pelo número total de notícias N_T que aconteceram numa transação durante o dia i, $\frac{\sum N^+}{N_T}$. Para o cálculo de notícias negativas *down trend*, o cálculo é o mesmo $\frac{\sum N^-}{N_T}$ e assim sucessivamente.

Após calcular a porcentagem total de notícias durante uma transação diária, se multiplica por um vetor α que contém os pesos das notícias. Para o cálculo de peso α nos baseamos em casos reais, já que nem todas as notícias têm a mesma importância. Existem milhares de notícias do tipo *non trend*, nesse caso o peso dado é baixo, as notícias *up trend* e *down trend* possuem um peso maior. A Equação 57, é parecida a anterior mas tem a diferença de calcular a classe com maior frequência numa transação diária.

Finalmente é adicionada o sinal do sentimento contida na transação Tr_i. Deste modo, uma transação diária da série temporal é considerada positiva, negativa ou neutra dependendo do número de notícias que aconteceram nela. A Figura 30 mostra um exemplo desta idéia do enriquecimento com o sentimento positivo durante um dia.

Figura 30 – Ilustração do processo de enriquecimento das séries temporais. É adicionado o sinal da média ponderada do sentimento com maior frequência nas características.

Date	Text	Label
2019-10-25	bitcoin price risk dangerous death cross ...	0
2019-10-25	will bitcoin death cross lead to another year ...	0
2019-10-25	johannesburg shut down computer city wide ...	0
2019-10-25	china president xi urge accelarate blockchain	1
2019-10-25	mining heavyweight partner with wef on blockchain	0
2019-10-25	bakkt bitcoin future set new record as bitcoin	-1
2019-10-25	simple cryptocurrency sell and buy strategy	0
2019-10-25	bitcoin president want to the country be global	1
2019-10-25	bitcoin bounce at key support level as bull ...	0

$$\begin{cases} = \left(\frac{\sum N^k}{N_T} \times \alpha[\mathbf{k}] \right) \\ = \left(\frac{\sum N^+}{N_T}, \frac{\sum N^-}{N_T}, \frac{\sum N^0}{N_T} \times \alpha[\mathbf{k}] \right) \\ = \left(\frac{2}{9} \times 0.9, \frac{1}{9} \times 0.9, \frac{6}{9} \times 0.1 \right) \\ = argmax\,(0.2, 0.1, 0.06) \\ = getClass\,(0.2) \\ = 1 \end{cases}$$

Fonte: Elaborada pelo autor.

7.6 Modelo de Previsão do Preço

Finalmente, o último passo de nossa proposta é a previsão do preço do Bitcoin. Nas anteriores seções foram explicadas os passos e os modelos construídos em cada um deles. O processo de mineração de texto extrai o conhecimento externo encontrado nas notícias. Esse conhecimento é utilizado para enriquecer a série temporal o alinhamento das notícias nas características compactas aprendidas pelo modelo de extração de representações compactas.

Para a previsão, nossa proposta utiliza vários *baseline* baseados em LSTM e comparamos com outra proposta, baseada na arquitetura *Dual Attention based Recurrent Neural Model (DA-RNN)* (QIN et al., 2017), explicada na literatura relacionada na Seção 6, esse modelo foi adaptado a nossas necessidades. Ele propõe enfocar-se em partes importantes da série temporal para extraí-las com um *encoder*, ou seja também utiliza características relacionadas a série temporal como indicadores técnicos. Após obter essas características relevantes ou *hidden state*, cria outro mecanismo de atenção para selecionar as relevantes *hidden state* sobe toda a série temporal, assim permite capturar adequadamente a longa dependência.

A maior parte dos trabalhos na literatura, para fazer a previsão de um número contínuo, utiliza um modelo de dependência longa, como a LSTM. Esse tipo de modelo melhora o desempenho de dependência longa, porém ainda não soluciona o problema. A diferença entre a LSTM e a DA-RNN é que o segundo codifica toda a sequência de entrada em um vetor de contexto, o mecanismo de atenção permite focar em diferentes partes da sequência de entrada para cada etapa da saída, ou seja, o *decoder* utiliza todos os estados intermediários para gerar a saída. Os resultados desta parte serão apresentados no experimento 13.

Capítulo 8

Resultados e Discussões

Nesta seção, conduzimos os experimentos, avaliamos a acurácia de cada algoritmo e nossa proposta comparada com outro algoritmo. Nosso trabalho esta seqüenciado em partes, a primeira parte descreve os *datasets* utilizados, a segunda parte descreve os protocolos de avaliação e finalmente a última parte descreve os experimentos.
Realizamos quatro experimentos na lista:

a) Experimento 1: o alvo neste experimento é conseguir um *dataset* rotulado para a classificação. Avaliamos a propagação do rótulo utilizando diferentes estratégias semissupervisionadas.

b) Experimento 2: avalia a acurácia da classificação dos textos comparando com outras técnicas de classificação de textos. Neste experimento se utilizou algoritmos baseados em *fine-tuning*.

c) Experimento 3: avalia o modelo de extração de características compactas, explicadas na Seção 7.4

d) Experimento 4: avalia o modelo da previsão do preço. Os resultados neste experimento estão relacionado a hipótese declarada nesta pesquisa. Primeiro avaliamos um modelo de previsão utilizando só o preço com um modelo que usa uma representação compacta de características. Finalmente avaliamos o impacto de enriquecer as séries temporais com dados textuais ou sem eles. Este experimento procura saber se os dados textuais ajudam a melhorar a previsão da série temporal em comparação a não utilizar dados textuais embutidos no modelo de previsão.

1. **Dataset**: para o estudo dos experimentos construímos dois *datasets*. 1) o primeiro conjunto de dados correspondente a notícias relacionadas ao *Bitcoin*. Este passo foi

explicado mais detalhadamente na Seção 7.3.1. O corpus coletado contém dados de dois anos de 1 de novembro de 2018 a 3 de novembro de 2020, que contém 10.111 notícias.

O segundo conjunto de dados contém os dados históricos da série temporal relacionadas ao *Bitcoin* obtidos a partir do *site Bitstamp*. O conjunto de dados foi coletado de 1 Setembro de 2015 a 3 Novembro de 2020, foi configurado para conter 6 observações de 4 horas por dia. Este conjunto de dados contém varias características geradas além dos preços, como foi explicado na Seção 7.4.1.

2. **Protocolo**: a avaliação dos resultados será realizada com medidas de avaliação de classificação e de regressão. Primeiro, avaliamos os modelos semissupervisionados empiricamente e utilizando a métrica acurácia. O segundo modelo avaliado é o modelo mineração de dados textuais, se utilizará as métricas tradicionais de classificação precisão, revocação, *f1-score* e a *matriz de confusão*.

   O desempenho do modelo de mineração de dados é avaliado pelo cálculo da matriz de confusão. A matriz de confusão é uma ferramenta útil para a análise de quão bom é um classificador na hora de classificar o número de predições corretas e incorretas em cada classe.

   O terceiro modelo avaliado é o modelo de geração de características compactas das séries temporais, se utilizaram as medidas de avaliação MAE, MSE, RMSE para a reconstrução das características da função *decoder* do Autoencoder. Finalmente, será avaliado o modelo de previsão do preço, neste modelo avaliaremos incluindo informação textual externa para enriquecer as séries temporais e sem essas informações, para isso será utilizado medidas de avaliação de regressão MAE, RMSE, MAPE.

8.1 Experimento 1: Propagação do Rótulo

Neste experimento, avaliamos a propagação do rótulo com o algoritmo semissupervisionado *Label Propagation* (LP) utilizando diferentes métodos de representação de textos (embeddings) no espaço vetorial. Além disso, fazemos uma comparação com outra estratégia para rotular, o algoritmo Node2Vec junto com o KMeans.

Previamente à aplicação do algoritmo LP, foi construído um grafo de nós e arestas utilizando as diferentes representações do texto. O grafo contém 10111 nós e 30334 arestas. Os nós representam as características dos textos e as arestas representam a similaridade entre nós. Cada nó contribui com a criação de, no máximo, 3 arestas. Foi utilizado a distância de similaridade de cosseno para criar as arestas entre nós. A similaridade miníma para a criação de uma aresta foi de 0.85 (1 significa muito similar e 0 indica nada similar).

Os rótulos para as classes foram: *up trend*, *non trend* e *down trend*. A classe *up trend* indica que um texto pode ter o potencial de impactar positivamente na série temporal. A classe *down trend* expressa um sentimento negativo que levaria ao mercado a uma queda nas próximas horas ou dias. Finalmente, a classe *non trend*, são notícias que não impactam o mercado consideradas como ruído ou seja não são nem positivas nem negativas tipo: informativas, análise de preços etc.

A Tabela 4, mostra o número dos rótulos para o conjunto inicial de notícias, obtido antes do processo de propagação explicado na Seção 7.3.3.

Tabela 4 – Configuração inicial dos rótulos no conjunto de notícias.

Rótulo	Quantidade
Up trend	80
Dow trend	135
Non trend	5979
Unlabeled	3917

Nesse conjunto, há milhares de notícias que estão relacionadas ao *bitcoin* direta ou indiretamente, mas não são suficientemente importantes como para desenvolver uma tendência ou movimentar os preços. Para este tipo de notícias, o modelo tem que ter a capacidade de identificar, mesmo não sendo importantes (no Experimento 8.4 veremos como ignorá-las), porém é importante classificá-las. Devido à natureza de nosso problema, há poucas notícias que têm o potencial de impactar o mercado positivamente ou negativamente.

A Tabela 5, mostra os resultados da propagação de rótulos utilizando diferentes estratégias de rotulado com diferentes representações dos textos. Por exemplo, a classe *up trend* contêm 80 notícias positivas rotuladas e é espalhado a 87; de 135 notícias *down trend* a 158 notícias utilizando a representação *TF-IDF* com 10000 características. Para comparar as técnicas foi calculado a acurácia com 30% dos textos rotulados reais da Tabela 4 e os nós preditos.

Foram testadas três estratégias: *LP+GFHF*, *LP+LLGC* e *Node2Vec+KMeans*. Pode-se observar que houve um incremento considerável de exemplos rotulados na classe *non trend*, de 5979 é espalhado a 9790, 9883 e 9881 utilizando a estrátegia *LP+GFHF* com as representações *TF-IDF*, *Word2vec embeddings* e *BERT embeddings* (utilizando 512 características), respectivamente. Quando se usa *TF-IDF* a acurácia cai a 0.94. O ganho maior se observa na classe *non trend* utilizando qualquer tipo de representações do textos, o número é alto, inclusive não existe muita diferença entre usar *LP+GFHF* e *LP+LLGC* já que contém quase o mesmo número de rótulos.

Pode-se concluir que não há grande diferença em usar algum *framework* com a topologia de nossa rede. Os dois frameworks conseguiram espalhar o rótulo, inclusive com a

Tabela 5 – Experimentos comparativos da rotulação semissupervisionada com o Label Propagation com as funções GFHF e LLGC. Para comparar se utilizou o Node2Vec

Representação	Up trend	Down trend	Non trend	Sem rótulo	Acurácia
LP + GFHF(iter=50)					
TF-IDF (10000f)	87	158	9790	76	0.940
word2vec (512f)	82	146	9883	0	0.963
BERT embedding (512f)	**84**	**146**	**9881**	0	0.962
LP + LLGC($\mu = 0.85$, iter=50)					
TF-IDF (10000f)	87	135	9828	61	0.946
word2vec (512f)	82	135	9886	0	0.965
BERT embedding (512f)	**83**	**135**	**9887**	0	0.965
Node2Vec(p=q=1) + KMeans (k=3)					
TF-IDF (10000f)	2197	2720	5102	0	0.540
word2vec (512f)	1892	3286	4933	0	0.951
BERT embedding (512f)	**3605**	**3275**	**3231**	0	0.963

estratégia LP+LLGC, que corrige algum tipo de erro em um nó possivelmente mal rotulado, ao utilizar $\mu = 0.85$ fazendo mais *soft* o espalhamento do rótulo aos vizinhos. Foram entre 50 interações do LP para garantir a convergência.

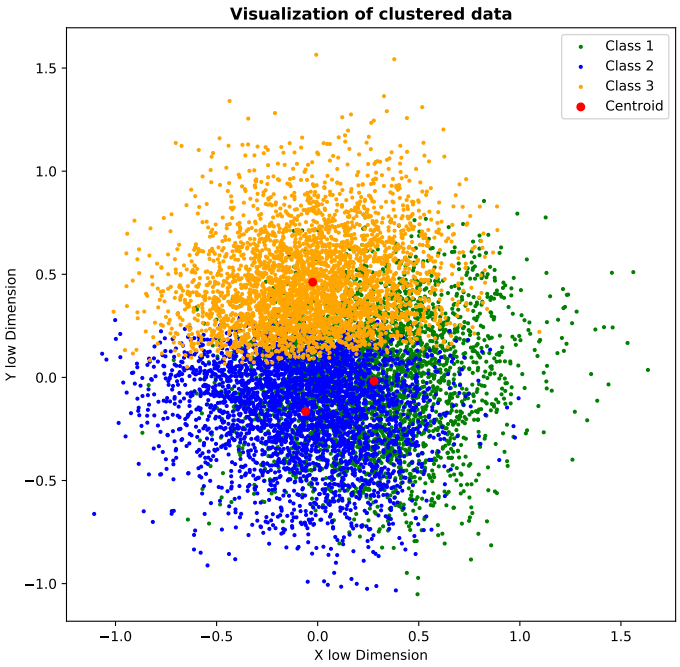
As duas funções do LP, de acordo com nossos experimentos, garantem um corpus anotado mais confiável com as notícias que realmente impactam o mercado mas acarretaram um problema no futuro para algum modelo de classificação, que é o "problema do desbalanceamento das classes". Esta estratégia serve para rotular textos e enriquecer uma série temporal em modo *off-line*. Quando falamos de *off-line* refere-se a não usar esta informação do rótulo num classificador em tempo real em produção.

Outra estratégia para a rotulação de textos semissupervisionado consiste na transformação da mesma rede ou grafo de notícias numa *embedding* de vetores representando nós de menor dimensão. Para isso, a mesma Tabela 5, mostra os resultados da combinação dos algoritmos *Node2Vec+KMeans*, explicados na Seção 5.5.3. Os hiper-parâmetros usados para o *Node2vec* foram: 3 dimensões, 10 caminhos por nó, longitude de cada nó 50, $p = q = 1$ (foram testados outros valores para p e q para explorar, mas tivemos melhores resultados com 1).

Com esta técnica conseguimos classes mais balanceadas, este tipo de estratégia consegue respeitar em outra dimensão a topologia da rede para que os pontos gerados sejam agrupados por o algoritmo KMeans. Por exemplo, as classes *up trend* e *down trend* estão mais balanceadas, com prevalência da classe *non trend* com qualquer representação, coisa que faz sentido. Quando utilizamos BERT Embeddings a acurácia é 0.963 maior que utilizar TF-IDF. Pode-se observar que tem muita diferencia entre dados rotulados com *Node2Vec+KMeans* e *LP+LLGC*. A Figura 31 mostra um *plot* de como foram divididos os *clusters* com o *KMeans*. Porém esta técnica não nos deu confiança numa inspeção visual

dos textos rotulados, por isso será descartada na avaliação da previsão no experimento final.

Figura 31 – Visualização dos pontos no espaço 2D, produzidos pelo Node2Vec. Foi aplicado o KMeans para encontrar $k = 3$ clusters.



Fonte: Elaborada pelo autor.

Como resultados indiretos a este experimento, utilizamos a série temporal como suporte para encontrar os períodos críticos de interesse nela, esses períodos servem para encontrar as notícias inicialmente rotuladas (*up-trend*, *non-trend*, *down-trend*) na Tabela 4, esta estratégia "Identificação de notícias críticas" foi explicada na Seção 7.3.3.

A Figura 32 mostra um *plot*, onde foram encontrados 32 períodos críticos. Por exemplo, se identificou muita volatilidade no período de 10 a 14 de março de 2020, que foram dias que a pandemia por covid19 afetou muitos mercados internacionais. Outro período de volatilidade aconteceram no 24 até 26 de Outubro do 2019, onde a China aceitou a tecnologia *Blockchain* no país. A Tabela 4 mostra o total de notícias encontradas nesses períodos.

A Figura 33, mostra um resultado visual da rotulação da rede de notícias, que contêm nós e arestas. Os nós contém as representações dos textos e as arestas indicam que existe similaridade entre textos. Pode-se observar como no segundo gráfico após rodar o algoritmo LP, o rótulo é espalhado aos seus vizinhos similares em cor verde e vermelho.

Figura 32 – Rotulando notícias críticas com apoio das séries temporais. As linhas azuis mostram os períodos de queda ou alta significativas.

Fonte: Elaborada pelo autor.

8.2 Experimento 2: Classificação e Fine-Tuning

O segundo experimento têm como objetivo avaliar a classificação dos textos. Neste experimento, após rotulados os textos no passo anterior e num formato entendível ao computador, pode ser aplicar algum tipo algoritmo para a tarefa de classificação. Utilizamos o conjunto de dados produzido pelas estratégias semissupervisionadas no experimento anterior. Mostramos resultados com diferentes algoritmos de aprendizado de máquina. Além disso, também fazemos uso modelos pré-treinados baseados em *BERT*, realizando a tarefa de *fine-tuning*.

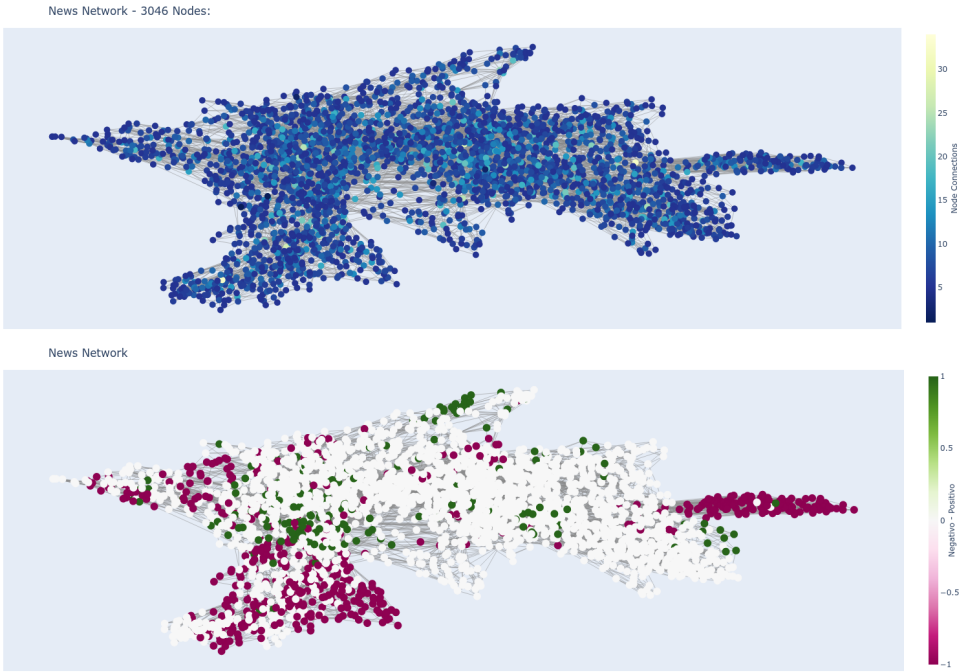
Antes do treinamento dos modelos, devido a natureza de nosso problema, o conjunto de dados é extremamente desbalanceado, os classificadores *baseline* foram treinados com os resultados (textos rotulados) da Tabela 5. O desbalanceamento é um problema típico em ambientes reais em comparação de dados sintéticos e controlados. Foi um desafio a mais lidar com um conjunto de dados com uma classe majoritária, uma vez que isso afeta o desempenho da classificação. A Figura 34 mostra o desbalanceamento das classes.

No nosso conjunto de dados existem poucas notícias *up trend* (+) e *down trend* (-1) e muitas notícias rotuladas como *non-trend* (0) que não influenciam o mercado mas precisam ser classificadas. Para lidar com o problema do desbalanceamento utilizamos técnicas como *random oversampling*, *weight class* e *focus loss* para tentar lidar com este problema.

Além da distribuição das classes, também analisamos a distribuição das palavras no conjunto de dados, como mostrado na Figura 35.

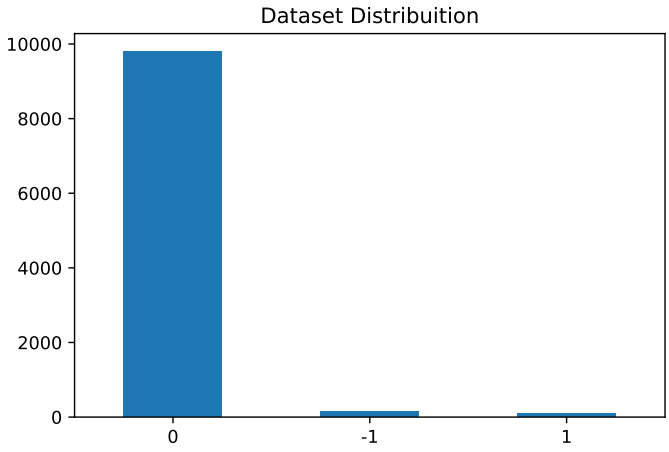
Para as Tabelas 6, 7, 8 foi utilizado o conjunto de dados rotulado pela estratégia LP-LLGC, e usamos classificadores *baseline*, com três tipos de representações. Foram utilizadas as representações *TF-IDF, Word2Vec e BERT Embeddings*. Experimentamos com essas últimas representações baseadas em *embeddings* para ver se existe alguma dife-

Figura 33 – Na figura acima, mostra-se uma rede de similaridade de notícias com nós sem rótulos e arestas. A figura abaixo, contém o mesma rede com seus rótulos espalhados pelo algoritmo *Label propagation* a partir de um conjunto inicial de rótulos. A cor verde representa o grupo de exemplos de notícias *up trend*, a cor vermelha, de notícias *down trend* e a cor branca representa notícias *non trend*.



Fonte: Elaborada pelo autor.

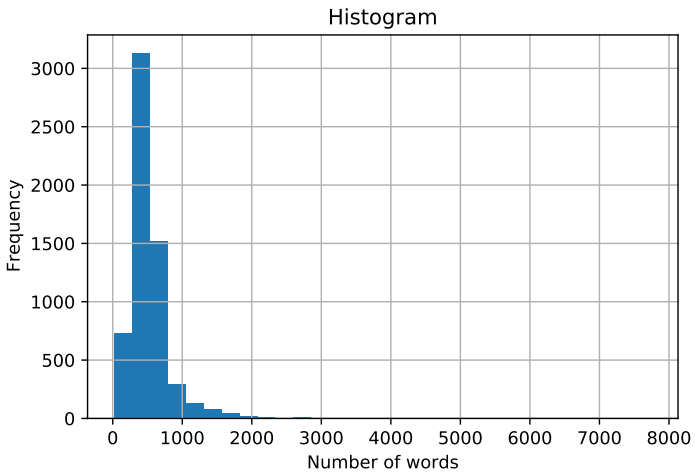
Figura 34 – Distribuição das classes.



Fonte: Elaborada pelo autor.

rença ao representar palavras, sobretudo no contexto semântico e sintático. O conjunto de dados foi dividido em 80% para treino, 10% para teste e 10% para validação. A Tabela 6 mostra os resultados do teste o *Logistic Regression* (LR), usando diferentes representações do texto.

Figura 35 – Histograma da distribuição das palavras.



Fonte: Elaborada pelo autor.

Tabela 6 – Precisão, Revocação, F1-score utilizando diferentes representações.

	Regressor Logístico		
Representação:	TF-IDF	Word2Vec	BERT Embeddings
	Classes	Classes	Classes
Acurácia	**96.59**	63.06	87.59
Precisão	**48.46**	**33.54**	**34.59**
Revocação	49.66	36.45	38.77
F1-Score	48.81	27.47	34.24

Pode-se observar que o LR tem uma acurácia alta porém não da confiança devido que a precisão, revocação muito baixos, menores que o 50% ou seja o modelo contém muito Falso Positive (FP) e Falso Negative (FN). Neste experimento se usou a *oversampling* para lidar com o desbalanceamento.

Tabela 7 – Precisão, Revocação, F1-score para todas as classes utilizando SVM.

	SVM		
Representação	TF-IDF	Word2Vec	BERT Embeddings
	Classes	Classes	Classes
Acurácia	98.01	89.12	97.27
Precisão	**83.00**	**34.71**	**45.59**
Revocação	39.56	38.90	37.89
F1-Score	43.78	34.64	39.41

Para a construção das representações de texto baseada em frequência de palavras, o *TF-IDF* foi construído com 10000 características. No pré-processamento não foram inclusos números nem *stopwords*. Para a construção das representações baseadas em *Word2Vec Embeddings* foi utilizada a dimensão de 512 para o vetor de cada palavra, a

estratégia Skingram e com o contexto de janela de 50 palavras.

Tabela 8 – Acurácia, Precisão, Revocação, F1-score utilizando Random Forest.

	Random Forest		
Representação	TF-IDF	Word2Vec	BERT Embeddings
	Classes	Classes	Classes
Acurácia	**98.01**	**97.67**	**97.69**
Precisão	**78.23**	**55.54**	**56.20**
Revocação	38.78	36.48	37.31
F1-Score	42.49	38.21	39.41

Para as representações baseadas em *BERT Embedding*, se utilizou o pacote *sentence-transformers* [1] com o modelo pré-treinado *distilBERT-base-nli-stsb-mean-tokens*, o mais adequado para calcular a similaridade do cosseno entre vetores, que recomendam os autores (REIMERS; GUREVYCH, 2019). Foi configurada a dimensão máxima de 512 para cada palavra. Além disso, para cada documento foi calculado seu vetor característico que melhor representa toda a sentença. As Tabelas 7 e 8 mostram os resultados para os classificadores SVM e Ramdom Forest (RF). Pode-se observar entre os classificadores *baseline* o RF tem melhor acurácia e precisão ao usar representações baseadas em *Embeddings* comparado com o LR e o SVM. Porém, a revocação ainda é baixa.

Como foi explicado na Seção 7.3.5, utilizamos o *BERT*, um modelo de contextos bidirecionais baseado em *Transformers* 4.4.3. O BERT pode ser usado como um classificador ou como um gerador de *Embeddings*, foram utilizados em ambas formas para nossos experimentos. Usamos o BERT-base *uncased* que contém 12 blocos de *transformers*, 12 self-attention heads e um vetor de 768 estados ocultos. A entrada do BERT são *Embeddings* com um máximo de 512 palavras. Nossa arquitetura esta baseada em *fine-tuned* sobe o modelo pré-treinado para especializá-lo e assim realizar a multi-classificação com três classes. Esta espacialização consiste em utilizar o modelo pré-treinado e utilizando apenas as últimas camadas para especializá-lo junto com uma base de dados pequena para a tarefa de classificação. Congelamos as últimas camadas e adiciona-se novas camadas ao modelo. Esta última camada, pode ser uma camada *softmax* com três saídas.

A arquitetura do *fine-tuning* é composta por: uma camada Dropout, uma função de ativação ReLU, uma camada Linear (768, 512), outra camada linear(512, 3) e uma função *softmax*. A função *Loss* usada para o otimizador foi a *NLLLoss* usado para classificar as três categorias.

Para o cálculo da taxa de aprendizado (*learning rate* do termo em inglês) se fez uma inspeção visual com diferentes taxas e épocas para encontrar a melhor combinação de hiper-parâmetros que se ajustam ao modelo e dados. Por exemplo, a Figura 36 (a) mostra que a função de custo do FastText se mantém estática com valores *learning rate* desde

[1] https://github.com/UKPLab/sentence-transformers é uma libraria com modelos pre-treinados prontos para diferentes casos.

$1e-7$ até $1e-4$, só converge a partir de $5e-4$ até $1e-2$. Já com valores maiores que $1e-2$, o algoritmo não converge e cai em *exploding*.

Figura 36 – Taxa de aprendizado para algoritmo FastText e o BERT.

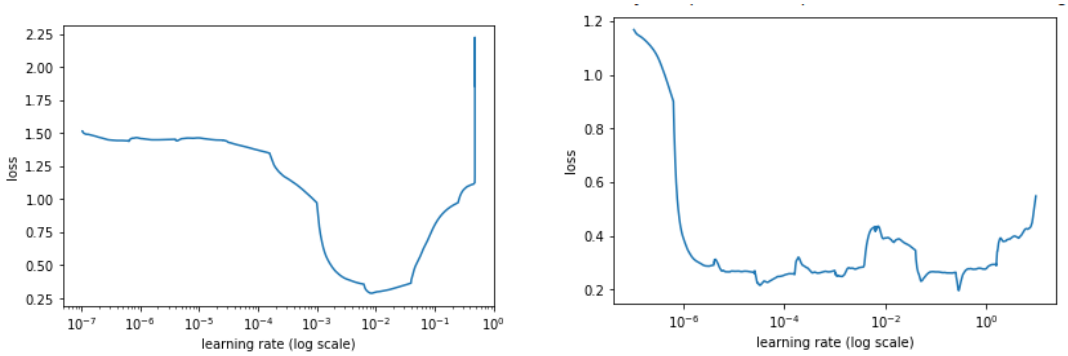


Figura (a) Figura (b)

Fonte: Elaborada pelo autor.

Tabela 9 – Métricas para a multi-classificação. Usando diferentes rotulado de textos.

	NB-SVM	FastText	BERT	DistilBERT
	Classes	Classes	Classes	Classes
Rotuladas por: Label Propagation				
Acurácia	88.49	93.00	97.00	98.00
Macro Precisão	39.38	48.00	32.00	33.00
Macro Revocação	40.95	37.00	33.00	33.00
Macro F1-Score	39.83	39.00	33.00	33.00
Rotuladas por: Node2Vec+KMeans				
Acurácia	78.00	79.00	**85.00**	**87.00**
Macro Precisão	79.00	79.00	**85.00**	**87.00**
Macro Revocação	78.00	80.00	**85.00**	**87.00**
Macro F1-Score	79.00	79.00	85.00	87.00

Finalmente, fazemos o estudo comparativo entre o modelo baseline NB-SVM (WANG; MANNING, 2012) e os modelos FastText (BOJANOWSKI et al., 2017), BERT e DistilBERT (SANH et al., 2020).

O DistilBERT é uma versão destilada do BERT, rápida, barata e *light*. Com ela é possível reduzir o tamanho do modelo em 40%, mantendo 97% da suas capacidades de entendimento da linguagem preservadas e ser 60% mais rápido no treinamento do modelo. Devido a que o BERT precisa de um entorno de hardware custoso (memoria e GPU) que as vezes não é acessível a muitos, o DistilBERT é uma excelente opção mais barata. Em termos de tempo, os modelos mais rápidos no treinamento foram o NB-SVM e o FastTex. Comparando o treinamento do BERT com esses modelos, na média demorou 6.30 horas sua variante DistilBERT demorou 25 minutos. Para os casos do BERT e DistilBERT se utilizou as versões pre-treinadas *BERT-base-uncase*.

A Tabela 9 mostra os resultados comparativos para esses algoritmos. No primeiro grupo, os textos rotulados pelo LP, claramente pode se observar que as métricas: precisão, revocação e f1-score são baixas (menor que 50%) e não conseguiram lidar bem com o desbalanceamento das classes, aprendendo muito da classe maioritária. Porém não significa que o algoritmo LP sejam ruim. Por outro lado, os dados rotulados pela estratégia *Node2Vec+KMeans* contém melhor acurácia e confiança a nível de métricas, por exemplo destaca os resultados do BERT e DistilBERT.

Para tentar lidar como o problema do desbalanceamento do LP, experimentamos duas técnicas: *Class Weight* e Focal Loss. A *Class Weight* consiste em um vetor de pesos que contém a frequência das classes maioritária e minoritárias, estes pesos permite que o modelo ajude a aprender mais as características dos exemplos menos representativos, dando mais peso às classes minoritárias. Esta técnica não alcançou muito impacto na melhora das métricas só aprendeu a classe maioritária.

A Segunda técnica, a Focal Loss (FL)(LIN et al., 2018), reescreve a função de custo *Loss Cross Entropy* a uma nova função $FL(p_t) = -(1-p_t)^{\lambda} log(p_t)$ para não aprender das classes que estão enviesadas, evitando a contribuição da classe maioritária a função *Loss*. O termo $(1-p_t)^{\lambda}$ é usado para melhorar a acurácia. Esta função FL foi usada no otimizador de BERT mas não conseguimos os resultados esperados, devido a implementações técnicas internas do Keras e o BERT. Como trabalho futuro de nosso projeto pretendemos explorar mais a fundo esta técnica para conseguir um classificador com boa acurácia, já que utilizar dados desbalanceados respeita a natureza de nosso problema em lugar de usar classes relativamente balanceadas que podem não refletir o impacto positivo ou negativo dos textos na série temporal.

As Tabela 11, 10, mostra a matriz de confusão para os dados preditos no conjunto do teste do BERT e DistilBERT.

		Verdadeiro		
		Down	Up	Non
Predito	Down	254	2	16
	Up	8	231	25
	Non	26	28	320

Tabela 10 – Matriz de confusão do DistilBERT, rotulados pelo Node2Vec+Kmeans

		Verdadeiro		
		Down	Up	Non
Predito	Down	231	6	22
	Up	7	217	42
	Non	27	37	321

Tabela 11 – Matriz de confusão do BERT, rotulados pelo Node2Vec+Kmeans

As conclusões finais para esses experimentos são:

1. Nos experimentos com algoritmos tradicionais, foram utilizadas técnicas de *over-sampling* para lidar com o desbalanceamento de classes. No entanto, a acurácia não melhora nesse caso.

2. A diagonal da Tabela 10 (TP e TN) para as três classes está razoavelmente equilibrada. O erro do tipo II associado à revocação para a classe down-trend é $revocação = TP/(TP + FN) = 254/(254 + 8 + 26) = 0.88$ e o erro do tipo I, o $precisão = TP/(TP + FP) = 254/(254 + 2 + 16) = 0.93$. O modelo DistilBERT erra mais nos Falsos Negativos e menos nos Falsos Positivos.

3. Dependendo da natureza do problema, e da estratégia de rotulação selecionada, se a estratégia deixa as classes extremamente desbalanceadas, é recomendável manter esse desbalanceamento para enriquecer a série temporal, para nosso caso se manteve a rotulação pelo LP. Se forem utilizados os textos rotulados em um classificador, é melhor usar os dados razoavelmente balanceados para conseguir bons valores de acurácia como foram mostrados na Tabela 10,11. Em ambos casos as duas estrategias têm que usar-se para avaliar o *forecast* do preço.

4. Para melhorar a acurácia da classificação usando textos rotulados pela estratégia LP, pretendemos explorar o uso de *Focal Loss* para enfocar o aprendizado nas classes menos representativas assim se conseguirá melhores resultados referente as métricas precisão e revocação. Consequentemente, o modelo não classificará falsos negativos como verdadeiros, focando-se em aprender mais características das classes menos representativas.

8.3 Experimento 3: Extração de Características Compactas

Este experimento tem como finalidade a extração de padrões úteis compactados de uma lista de características explicados na Seção 7.4. Além de compactar as características, também avaliamos a precisão da reconstrução pelo modelo *Decoder*. Nossa finalidade na reconstrução é medir quão bom o *Decoder* consegue reconstruir a camada compacta a os dados originais. Esta camada compacta será o novo *input* para o cálculo da previsão do preço do próximo experimento. A arquitetura do modelo foi explicada na Seção 7.4.3.

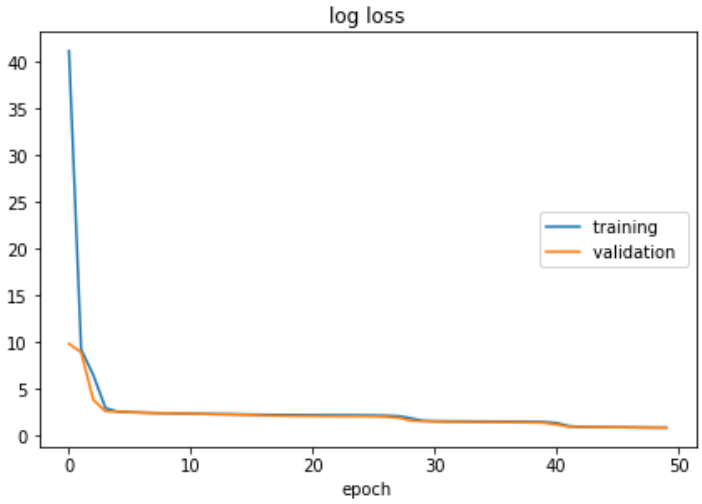
A Tabela 12, mostra as principais medidas de avaliação de regressão utilizadas para medir o erro do modelo na reconstrução aos valores originais. Por exemplo, utilizando 50 épocas se conseguiu um *MAE, MSE e RMSE* baixos, comparado os valores originais com os preditos. O coeficiente de determinação R^2 é alto (entre $0-1$), esta métrica representa quão bem os valores se ajustam em comparação com os valores originais. Quanto maior

Tabela 12 – Métricas do Teste de reconstrução das características com Autoencoder.

Hiper-parâmetros	lr=1e-3 e=15	lr=5e-4 e=30	lr=5e-3 e=50
R^2	0.764	0.771	**0.90**
MAE	0.033	0.032	0.021
MSE	0.003	0.003	0.001
RMSE	0.055	0.054	**0.031**

for o valor, melhor será o modelo. A Figura 37, mostra como a função de custo (Mean Square Error) converge durante o treinamento.

Figura 37 – Convergência da função de custo no treinamento do Autoencoder.



Fonte: Elaborada pelo autor.

8.4 Experimento 4: Previsão das Séries Temporais com Texto Enriquecido.

Finalmente, o último experimento é a previsão do preço. O objetivo final deste experimento é avaliar o impacto de incorporar as notícias no modelo de previsão em comparação com outros métodos de previsão de séries temporais. Também avaliamos o impacto de incluir as características mais representativas abstratas nos modelos construídas na seção anterior.

Para este experimento, previamente necessitamos fazer o processo de concatenação, explicado na Seção 7.5. Para a concatenação foram testados dois tipos de estratégias: a média ponderada das notícias numa transação e a máxima classe representativa numa transação. Os melhores resultados foram com a máxima classe representativa, definida

Tabela 13 – Resultado comparativo do erro da previsão das séries temporais enriquecidas com dados textuais utilizando o Label Propagation(off-line). O modelo DA-RNN+SE* foi enriquecido pela estratégia Node2Vec+KMeans.

Modelo	Sentimento	R^2	MAE	RMSE	MAPE (%)
LSTM	Não	0.934	0.195	0.257	34.10
LSTM+ENC	Não	**0.900**	0.072	0.098	**6.934**
LSTM+ENC+SE	Sim	0.777	**0.034**	**0.043**	7.13
DA-RNN(QIN et al., 2017)	Não	0.945	0.016	0.021	3.622
DA-RNN+SE*	Sim	0.978	0.010	0.013	2.166
DA-RNN+SE	Sim	**0.982**	**0.009**	**0.012**	**2.006**

pela Equação 58:

$$Tr_i = argmax_{k\in\{+,-,0\}} \left(\frac{\sum N^k}{N_T} \times \alpha[\mathbf{k}] \right) \tag{58}$$

O valor α é um vetor de pesos que permite configurar a importância dos tipos de notícias. Para as notícias mais importantes *up-trend(+)* e *dow-trend(-)* têm um peso de $\alpha = 0.90$, e para notícias do tipo *non-trend(0)* tem um peso de $\alpha = 0.1$, este parâmetro α varia entre os valores $[0, 1]$. Quando o valor $\alpha = 0$ anulamos o efeito das notícias, quer dizer que não é utilizado o conhecimento externo.

Desta forma, conseguimos enriquecer uma transação Tr_i diária com informação textual que foi rotulada pela estratégia usando o LP *off-line* ou Node2Vec+KMeans *on-line*. A Figura 38, mostra a densidade da coluna sentimento concatenada nas séries temporais, claramente pode-se observar três grupos, onde existem mais transações Tr_i com sentimento 0 e menos com sentimento -1 e 1, esta característica acreditamos, que permite ao modelo de previsão aprender informações textuais.

A Tabela 13, mostra uma visão geral dos resultados experimentais do processo de enriquecimento das séries temporais, com as melhores configurações dos modelos. Os dados foram divididos em treino 80% e teste 20%. O horizonte de previsão foi de $h = \{1, 3, 10\}$ usamos $h = 10$. Foi observado quando o h cresce a erro é maior. A LSTM foi configurada com uma janela deslizante de 30 observações passadas para prever o preço da observação $t + h$. O sentimento é expressado pelo valor $\alpha = \text{Sim} = 0.9$ e $\alpha = \text{Não} = 0.0$

O primeiro modelo é uma *baseline* LSTM, que faz a previsão utilizando valores históricos do preço (*univariate* ou seja utiliza somente uma característica o preço), este modelo não utiliza outras características, nem o sentimento. Quando avaliamos este modelo teve os piores resultados em termos das métricas RMSE e MAPE com 0.257 e 34.10.

A LSTM+ENC (Encoder) é parecida a anterior LSTM mas foi treinada diferente, ela recebe como *input* a saída do experimento 8.3, que contém as novas características aprendidas com o *AutoEncoder* de uma lista de indicadores técnicos mostrados na Tabela 3. Esta nova camada de características compactas aprendidas são de menor tamanho, para nosso caso utilizamos com 4 características compactas. Este modelo consegue me-

Figura 38 – O sentimento concatenado nas Séries Temporais. A figura a) mostra o sentimento produzido pela estrategia Label propagation, a figura b) utilizou a estrategia Node2Vec+Kmeans.

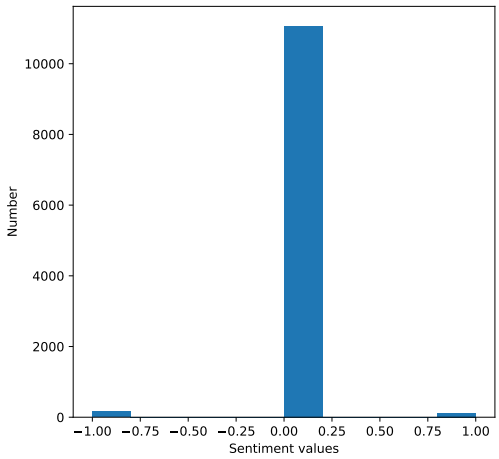


Figura (a)

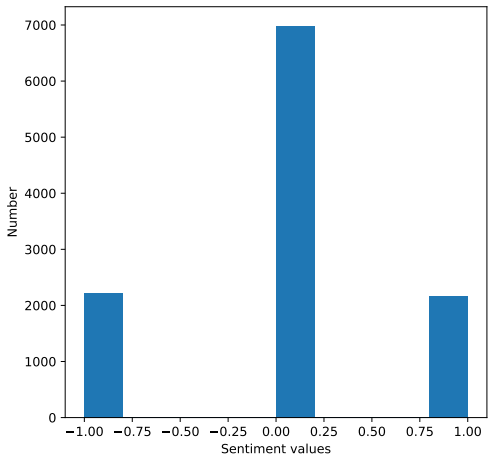


Figura (b)

Fonte: Elaborada pelo autor.

lhor acurácia em termos de MAE, RMSE e MAPE em comparação com a LSTM. A LSTM+ENC+SE (SE refere-se ao sentimento incorporado) contém o sentimento, igual que a anterior foi treinada com o AutoEncoder e o sentimento foi concatenado junto as características compactas. Este modelo consegue melhor acurácia em termos de MAE e RMSE mais foi ruim no MAPE.

Para comparar utilizamos a modelo DA-RNN proposto em Qin et al. (2017). A DA-RNN é um modelo que põe atenção nas características de um série temporal e em cada observação de toda série temporal no tempo. Este modelo contém: dois mecanismos de atenção com um *encoder*, um mecanismo de atenção e uma LSTM. O *encoder* extrai as características mais relevantes dos indicadores técnicos, a atenção extrai as partes mais

relevantes da série temporal no tempo e com estas duas informações servem como o novo *input* para a LSTM que faz a previsão.

Modificamos a DA-RNN e chamamos DA-RNN+SE contendo o sentimento. O input é composto pelos indicadores técnicos e o sentimento. A DA-RNN não considera informações do tipo sentimento. A configuração dos estados ocultos da DA-RNN+SE, do encoder (m) e o decoder (p) foram igualados por facilidade $m = p$, $p \in \{32, 64\}$, o horizonte de previsão h=10 observações e número de épocas entre 60-100.
As conclusões deste experimento são:

1. Com este experimento permitiu demonstrar que o modelo LSTM-ENC que extrai uma camada compacta dos indicadores técnicos mais relevantes de uma série temporal, e que utilizados como um novo input, o modelo de previsão consegue uma redução das métricas MAE, RMSE e sobre to o MAPE, se comparado com um modelo que não usa.

   Se comparamos a LSTM e a LSTM+ENC consegue uma redução do erro em 15.9% e 27% em termos de RMSE e MAPE, podemos afirmar a hipótese declarada no início deste trabalho, que utilizar uma representação abstrata de indicadores técnicos ou a Analise técnica melhora a acurácia.

2. Também podemos sugerir que ao adicionar informação do contexto processada por as etapas da mineração de sentimentos e seguindo o devido passos para a concatenação dessa informação nas séries temporais, a precisão da acurácia melhora para o caso do Bitcoin, em 3.7%, 1% e 1.62% em termos de R^2, RMSE e MAPE se for comparado a e DA-RNN+SE e a DA-RNN. Com esta melhora nos permite confiar na informação externa em forma de notícias considerada como ferramenta de um Análise fundamentalista.

3. Podemos confirmar a declaração completa da hipótese desta pesquisa. Que a informação externa e os indicadores técnicos ajudam muito na análise da série temporal aplicado nosso dominio.

Capítulo 9

Conclusões e Trabalhos Futuros

Neste capítulo apresenta-se as conclusões, limitações e trabalhos futuros desta pesquisa.

9.1 Conclusões

O trabalho descrito nesta dissertação teve como objetivo construir um modelo híbrido que permita adicionar o conhecimento externo de notícias nas séries temporais sem deixar de considerar os padrões passados da série temporal, com a finalidade de melhorar a previsão do preço do ativo digital *Bitcoin*.

Para a realização deste trabalho foi necessário várias etapas em cada uma nos encontramos com desafios e problemas a ser resolvidos. A primeira foi construir um corpus anotado, desde o início da construção do corpus até seu uso na inferência com os classificadores, tivemos que passar por vários processos. A coleta foi feita sobre uma grande quantidade de notícias produzidas a diário por vários sítios, estas notícias careciam de rótulo, poucos foram os textos anotados e os problemas enfrentados como o tempo que custa rotular, os recursos e os anotadores. Estes motivos nos permitiu desenvolver uma ferramenta que permita utilizar as séries temporais como suporte para encontrar notícias críticas por similaridade em momentos de alta ou queda.

Com os textos coletados e sem rotular ainda, realizamos um pré-processamento para a representação desses textos. Utilizamos diferentes técnicas de representação como: TF-IDF e Embeddings. Foi gerado uma *embedding* baseada em Word2Vec relacionada as notícias financeiras do Bitcoin como um aporte de nosso trabalho.

Para poder enriquecer as séries temporais, foi preciso um corpus rotulado para nossos propósitos. A grande quantidade de textos, foram um dos motivos de explorar uma

estratégia semissupervisionada que permita rotular com um grau de confiança. A estratégia escolhida foi o algoritmo *Label Propagation*, este nos permitiu espalhar o rótulo a mais textos. Embora, a quantidade textos rotulados pelo *Label Propagation* não tem sido promissórias, se comparados com outras estratégias de rotulado de textos semissupervisionadas, esta estratégia da mais confiança se é utilizada num contexto *off-line*, onde não se precise de um classificador, pode facilmente enriquecer as séries temporais.

O Bert e DistilBert não têm bons resultados se são treinados com um dataset com classes desbalanceadas que foram rotulados pelo algoritmo *Label Propagation*. Embora se utilize algumas técnicas de balanceamento não conseguimos generalizar bem quando os exemplos de uma classe são minoritários e foi enviesado com a classe maioritária. O Bert mostrou boa acurácia sobre todo em métricas f1-score e precisão com textos rotulados com outra estratégia semissupervisionada que comparamos Node2Vec e KMeans, era esperado. Mas esta estratégia de rotulado não apresenta confiança, já que não garante que aqueles textos pertençam realmente a classe. Este problema acreditamos que é devido ao problema de saber qual é o ótimo número de clusters KMeans e a distância muito pequena entre clusters.

Outro desafio encontrado no decorrer desta pesquisa foi a grande dissimilaridade entre textos, o que nos motiva a pensar que existem mais de três classes. Nossa pesquisa considera só três classes. É necessário explorar técnicas de clustering que permita encontrar que grupos existem no corpus.

Foi treinada uma rede neural LSTM-ENC utilizando características abstratas aprendidas de forma não supervisionada com o intuito de aprender as melhoras ferramentas da análise técnica. A acurácia da LSTM-ENC comparada com a LSTM que usa somente observações passadas do preço foi superior. Podemos concluir que é sumamente importante considerar os indicadores técnicos na análise da serie temporal.

Os resultados obtidos nos experimentos deste trabalho, nos permite confirmar a declaração mencionada na hipótese perseguida em nossa pesquisa. Sugerimos que a incorporação de notícias externas e padrões passados dos indicadores técnicos na análise da previsão, consegue reduzir o erro RMSE em 21.4% e MAPE em 26.97% se comparamos a LSTM e a LSTM+ENC+SE. Se comparamos a DA-RNN e DA-RNN+SE conseguimos a redução do erro em 3.7% e $1,62\%$ em termos de R^2 e MAPE.

9.2 Limitações e Trabalhos Futuros

A abordagem proposta permitiu reduzir o erro se incorporamos notícias no cálculo da previsão da série temporal. Embora isso seja um indicativo que o conhecimento externo possa ser útil, ainda há limitações nos experimentos durante o desenvolvimento desta pesquisa que são diretrizes para iniciar trabalhos futuros. Entre eles temos:

a) É um erro pensar que um algoritmo como o Bert, DistilBert ou outro, vá a ser capaz de identificar toda uma estrutura gramatical numa sentença, assumindo que em um representação no espaço vetorial o algoritmo seja capaz de identificar e fazer uma análise detalhada da sentença, identificando as partes *Part of Speach* (POS). Nossa proposta esta limitada pela incapacidade de fazer uma análise mais profunda do texto e responder outro tipo de questões como: "Onde?", "Que?", "Quando?".

   É importante a extração mais detalhada de entidades nomeadas e metadados: nomes de organizações, informação temporal, lugares, sobre quem atua a ação. Por exemplo a extração de entidades nomeadas, pode servir muito para saber de que entidade se esta falando e sobre que entidade acontece uma ação. Por exemplo: a companhia "A" fez uma parceria com a companhia "B". Não podemos saber sobre que entidade acontece o impacto em consequência que séries temporais serão enriquecidas da companhia "A" ou companhia "B". Pior ainda, a notícia pode ser considerada neutra para a companhia "A" e muito boa para a companhia "B", o sistema dará um falso positivo.

b) O Label Propagation é um algoritmo simples e ideal para rotular dados, porém há casos que ele deixa o corpus desbalanceado. Ele serve para rotular textos *off-line* com esses textos podem enriquecer as séries temporais, mas se quiser utilizar em um sistema em tempo real, o dataset precisa estar razoavelmente balanceado para classificar novos exemplos. O experimento comparativo Node2Vec+KMeans identificar três grupos com k=3, porém não se garante que aqueles grupos contenham notícias que realmente impactaram o mercado, motivo pelo qual se precisa estudar mais afundo outras estratégias de rotulado não supervisionado que consiga identificar o k ideal, para não forçar o KMeans a pertencer a um grupo.

   Existem outras estratégias para a propagação do rótulo na literatura da área de Networks Embeddings como: Metapath (CAO et al., 2017), Graph Neural Networks (ZHOU et al., 2019; CAI et al., 2020) que podem ser úteis para resolver este problema.

c) Realizar uma descoberta de tópicos de maneira não supervisionada sobre o corpus construído. Acreditamos que existem mais de três classes (up trend, down trend e non-trend). Forçar a todos os textos a pertencer a estes grupos pode ser indicativo de que o modelo não consiga diferenciar as notícias. Por exemplo queremos saber

de que tópicos se está falando para saber que classes existem e quais impactariam o mercado do Bitcoin.

d) Por ser uma pesquisa de mestrado, com tempo limitado e o tempo dedicado a experimentos em um entorno simulado, embora que nossos experimentos usam dados reais e estão mais parecidos ao mundo real no mercado do Bitcoin. Nossa pesquisa carece de uma estratégia de compra e venda utilizando nossa própria proposta que mostre os retornos. Como trabalho futuro se vê necessário implementar uma estratégia que permita avaliar o retorno e perdas, num sistema em produção com dinheiro real. Com esta métrica de retorno, seria um critério adicional para avaliar o modelo. Assim ficaria mais evidente como se evita o risco quando se expõe num mercado muito volátil como o Bitcoin.

Referências

ADHIKARI, R.; AGRAWAL, R. K. An introductory study on time series modeling and forecasting. **arXiv preprint arXiv:1302.6613**, 2013.

_____. A combination of artificial neural network and random walk models for financial time series forecasting. **Neural Computing and Applications**, v. 24, n. 6, p. 1441–1449, May 2014. ISSN 1433-3058.

AKITA, R. et al. Deep learning for stock prediction using numerical and textual information. In: **2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)**. [S.l.: s.n.], 2016. p. 1–6.

ALDRIDGE, I. **High-Frequency Trading: A Practical Guide to Algorithmic Strategies and Trading Systems**. [S.l.]: Wiley, 2009. (Wiley Trading). ISBN 9780470579770.

ANTONIO, C. G. **Métodos e técnicas de pesquisa social 6ta**. [S.l.]: Editorial Atlas S.A, 2008. 16 p.

ARCE, P. **Online learning methods for financial time series forecasting**. Tese (Doutorado) — University of Valparaiso, 2017.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2015.

BAO, W.; YUE, J.; RAO, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. **PLOS ONE**, Public Library of Science, v. 12, n. 7, p. 1–24, 07 2017.

BENABDALLAH, A.; ABDERRAHIM, M. A.; ABDERRAHIM, M. E.-A. Extraction of terms and semantic relationships from arabic texts for automatic construction of an ontology. **International Journal of Speech Technology**, v. 20, n. 2, p. 289–296, Jun 2017.

BENGIO, Y. et al. A neural probabilistic language model. **Journal of machine learning research**, v. 3, n. Feb, p. 1137–1155, 2003.

BENGIO, Y.; LECUN, Y. Scaling learning algorithms towards ai. In: _____. **Large-scale kernel machines**. [S.l.]: MIT Press, 2007.

Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. **IEEE Transactions on Neural Networks**, v. 5, n. 2, p. 157–166, March 1994. ISSN 1045-9227.

BHATTACHARYYA, G. K. 5 tests of randomness against trend or serial correlations. In: **Nonparametric Methods**. [S.l.]: Elsevier, 1984, (Handbook of Statistics, v. 4). p. 89 – 111.

BIESIALSKA, K.; BIESIALSKA, M.; RYBINSKI. Sentiment analysis with contextual embeddings and self-attention. **Foundations of Intelligent Systems**, Springer International Publishing, p. 32–41, 2020. ISSN 1611-3349. Disponível em: <http://dx.doi.org/10.1007/978-3-030-59491-6_4>.

BOJANOWSKI, P. et al. **Enriching Word Vectors with Subword Information**. 2017.

BOLLEN, J.; MAO, H.; ZENG, X. Twitter mood predicts the stock market. **Journal of Computational Science**, v. 2, n. 1, p. 1 – 8, 2011. ISSN 1877-7503.

BOUCHACHIA, A.; BOUCHACHIA, S. **Ensemble learning for time series prediction**. [S.l.]: na, 2008.

BRAUDEL, F.; REYNOLD, S. **Civilization and Capitalism, 15th-18th Century, Vol. I: The Structure of Everyday Life**. [S.l.]: University of California Press, 1992. (Civilisation matérielle, économie et capitalisme). ISBN 9780520081147.

BRUCE, R. A Bayesian Approach to Semi-Supervised Learning. **Nlprs2001**, p. 57–64, 2001.

CAI, L. et al. **Line Graph Neural Networks for Link Prediction**. 2020.

CAO, X. et al. Meta-path-based link prediction in schema-rich heterogeneous information network. **International Journal of Data Science and Analytics**, v. 3, n. 4, p. 285–296, Jun 2017. ISSN 2364-4168. Disponível em: <https://doi.org/10.1007/s41060-017-0046-1>.

CAVALCANTE, R. C. et al. Computational intelligence and financial markets: A survey and future directions. **Expert Systems with Applications**, v. 55, 2016. ISSN 0957-4174.

CERRI, R. **Redes neurais e algoritmos genéticos para problemas de classificação hierárquica multirrótulo.** São Carlos, Universidade de São Paulo: Instituto de Ciências Matemáticas e de Computação, 2013.

Chen, S.; Wang, X. X.; Harris, C. J. Narx-based nonlinear system identification using orthogonal least squares basis hunting. **IEEE Transactions on Control Systems Technology**, v. 16, n. 1, p. 78–84, Jan 2008.

CHEN, Y.; HAO, Y. A feature weighted support vector machine and k-nearest neighbor algorithm for stock market indices prediction. **Expert Systems with Applications**, v. 80, p. 340 – 355, 2017. ISSN 0957-4174.

CHO, K. et al. **Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation**. 2014.

CHRISTIE, P.; ISIDORE, R. Fundamental analysis versus technical analysis-a comparative review. **International Journal of Recent Scientific Research**, v. 9, p. 23009–23013, 01 2018.

CHUNG, J. et al. **Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling**. 2014.

CORRÊA, G. N. et al. Uso da mineração de textos na análise exploratória de artigos científicos. São Carlos, SP, Brasil., 2012.

Deng, L.; Yu, D. **Deep Learning: Methods and Applications**. [S.l.]: now, 2014.

DENG, S. et al. Deep learning for knowledge-driven ontology stream prediction. In: **Knowledge Graph and Semantic Computing. Knowledge Computing and Language Understanding**. Singapore: Springer Singapore, 2019. p. 52–64.

DEVLIN, J. et al. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. 2019.

DING, X. et al. Deep learning for event-driven stock prediction. In: **Twenty-Fourth International Joint Conference on Artificial Intelligence**. [S.l.: s.n.], 2015.

DOSCIATTI, M. M.; FERREIRA, L. P. C.; PARAISO, E. C. Anotando um corpus de notícias para a análise de sentimentos: um relato de experiência (annotating a corpus of news for sentiment analysis: An experience report). In: **Proceedings of the 10th Brazilian Symposium in Information and Human Language Technology**. [S.l.: s.n.], 2015. p. 121–130.

ELMAN, J. L. Finding structure in time. **Cognitive science**, Wiley Online Library, v. 14, n. 2, p. 179–211, 1990.

FACELI, K. et al. **Inteligência Artificial: Uma abordagem de aprendizado de máquina**. [S.l.]: Grupo Gen-LTC, 2000.

FELDMAN, R.; SANGER, J. **The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data**. [S.l.]: Cambridge University Press, 2006.

FISCHER, T.; KRAUSS, C. Deep learning with long short-term memory networks for financial market predictions. **European Journal of Operational Research**, v. 270, n. 2, p. 654 – 669, 2017. ISSN 0377-2217.

FREUND, Y.; HAUSSLER, D. Unsupervised learning of distributions on binary vectors using two layer networks. In: MOODY, J. E.; HANSON, S. J.; LIPPMANN, R. P. (Ed.). **Advances in Neural Information Processing Systems 4**. [S.l.]: Morgan-Kaufmann, 1992. p. 912–919.

FRIGOLA, R.; RASMUSSEN, C. E. Integrated pre-processing for bayesian nonlinear system identification with gaussian processes. In: IEEE. **52nd IEEE Conference on Decision and Control**. [S.l.], 2014. p. 5371–5376.

GAL, Y. **Uncertainty in Deep Learning**. Tese (Doutorado) — University of Cambridge, 2016.

GOLDBERG, Y. **A Primer on Neural Network Models for Natural Language Processing**. 2016.

GOMES, D.; EVSUKOFF, A. **Processamento de linguagem natural em Português e aprendizagem profunda para o domínio de Óleo e Gás**. 2019.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GRAHAM, B.; DODD, D. **Security analysis**. [S.l.]: Whittlesey house, McGraw-Hill book company, inc., 1934.

GRANGER, C. W. Investigating causal relations by econometric models and cross-spectral methods. **Econometrica: Journal of the Econometric Society**, JSTOR, p. 424–438, 1969.

GROVER, A.; LESKOVEC, J. **node2vec: Scalable Feature Learning for Networks**. 2016.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. Third edition. Boston: Morgan Kaufmann, 2012. (The Morgan Kaufmann Series in Data Management Systems).

HARRIS, Z. S. Distributional structure. **Word**, Taylor & Francis, v. 10, n. 2-3, p. 146–162, 1954.

HARTMANN, N. et al. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. **CoRR**, 2017.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. 2st. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1994. ISBN 0023527617.

HOCHREITER, S. Untersuchungen zu dynamischen neuronalen netzen. **Diploma, Technische Universität München**, v. 91, n. 1, 1991.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HOVY, E.; LAVID, J. Towards a 'science'of corpus annotation: a new methodological challenge for corpus linguistics. **International journal of translation**, v. 22, n. 1, p. 13–36, 2010.

HOWARD, J.; RUDER, S. **Universal Language Model Fine-tuning for Text Classification**. 2018.

HSIEH, T.-J.; HSIAO, H.-F.; YEH, W.-C. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. **Applied soft computing**, Elsevier, v. 11, n. 2, p. 2510–2525, 2011.

HUTTO, C.; GILBERT, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. **Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014**, p. 216–225, 01 2014.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting: principles and practice**. OTexts: Melbourne, Austrália, 2018. Disponível em: <http://OTexts.com/fpp2>.

JAIN, A. K.; DUBES, R. C. **Algorithms for Clustering Data**. USA: Prentice-Hall, Inc., 1988. ISBN 013022278X.

JASEMI, M.; KIMIAGARI, A. M.; MEMARIANI, A. A modern neural network model to do stock market timing on the basis of the ancient investment technique of japanese candlestick. **Expert Systems with Applications**, v. 38, n. 4, p. 3884 – 3890, 2011. ISSN 0957-4174.

JOTHIMANI, D.; SHANKAR, R.; YADAV, S. S. Discrete wavelet transform-based prediction of stock index: A study on national stock exchange fifty index. **Journal of Financial Management and Analysis**, v. 49, p. 22–23, 2016.

KENDALL, A.; GAL, Y. What uncertainties do we need in bayesian deep learning for computer vision? In: **Advances in Neural Information Processing Systems 30 (NIPS)**. [S.l.: s.n.], 2017.

KIM, B. **Introduction to Attention Mechanism**. 2020. <https://buomsoo-kim.github.io/attention/2020/01/01/Attention-mechanism-1.md/>.

KIM, Y. B. et al. Predicting fluctuations in cryptocurrency transactions based on user comments and replies. **PLOS ONE**, Public Library of Science, v. 11, n. 8, p. 1–17, 08 2016.

KIRKPATRICK, C.; DAHLQUIST, J. **Technical Analysis: The Complete Resource for Financial Market Technicians**. [S.l.]: Pearson Education, 2010. ISBN 9780132599627.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. 2007.

KOPPEL, M.; SHTRIMBERG, I. Good news or bad news? let the market decide. In: _____. [S.l.: s.n.], 2004. v. 20, p. 297–301.

KUMAR, D. A.; MURUGAN, S. Performance analysis of indian stock market index using neural network time series model. In: **2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering**. [S.l.: s.n.], 2013. p. 72–78.

LAI, G. et al. **Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks**. 2018.

LAI, S. et al. Recurrent convolutional neural networks for text classification. In: **Twenty-ninth AAAI conference on artificial intelligence**. [S.l.: s.n.], 2015.

LAKATOS, E. M.; MARCONI, M. A. **Fundamentos da metodologia científica 5ta**. [S.l.]: São Paulo: Atlas, 2003.

LAPTEV, N. et al. Time-series extreme event forecasting with neural networks at uber. In: . [S.l.: s.n.], 2017.

LAZAR, J.; FENG, J. H.; HOCHHEISER, H. **Research methods in human-computer interaction**. [S.l.]: Morgan Kaufmann, 2017.

LEANDRO, J. et al. Introdução ao Processamento de Linguagem Natural usando Python. **III Escola Regional de Informática do Piauí**, p. 336–360, 2017.

LI, Y. et al. Learning word representations for sentiment analysis. **Cognitive Computation**, Springer, v. 9, n. 6, p. 843–851, 2017.

Lien Minh, D. et al. Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network. **IEEE Access**, v. 6, p. 55392–55404, 2018.

LIN, T.-Y. et al. **Focal Loss for Dense Object Detection**. 2018.

LIU, B. Sentiment analysis and opinion mining. **Synthesis lectures on human language technologies**, Morgan & Claypool Publishers, v. 5, n. 1, p. 1–167, 2012.

_____. **Sentiment analysis: Mining opinions, sentiments, and emotions**. [S.l.]: Cambridge University Press, 2015.

LIU, W. et al. A survey of deep neural network architectures and their applications. **Neurocomputing**, v. 234, p. 11 – 26, 2017. ISSN 0925-2312.

MAKRIDAKIS, S.; HIBON, M. The m3-competition: results, conclusions and implications. **International journal of forecasting**, Elsevier, v. 16, n. 4, p. 451–476, 2000.

MALKIEL, B. G.; FAMA, E. F. Efficient capital markets: A review of theory and empirical work. **The journal of Finance**, Wiley Online Library, v. 25, n. 2, p. 383–417, 1970.

Marcacini, R. M.; Carnevali, J. C.; Domingos, J. On combining websensors and dtw distance for knn time series forecasting. In: **2016 23rd International Conference on Pattern Recognition (ICPR)**. [S.l.: s.n.], 2016. p. 2521–2525.

Martinez, L. C. et al. From an artificial neural network to a stock market day-trading system: A case study on the bm amp;f bovespa. In: **2009 International Joint Conference on Neural Networks**. [S.l.: s.n.], 2009. p. 2006–2013. ISSN 2161-4393.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

MELLO, R. F. de; PONTI, M. A. **Machine Learning: A Practical Approach on the Statistical Learning Theory**. [S.l.]: Springer, 2018.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. 2013.

_____. Distributed representations of words and phrases and their compositionality. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2013. p. 3111–3119.

MITCHELL, T. M. **Machine Learning**. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072.

MONTGOMERY, D. C.; JENNINGS, C. L.; KULAHCI, M. **Introduction to time series analysis and forecasting**. [S.l.]: John Wiley and Sons, 2015.

MURPHY, J. J. **Technical analysis of the financial markets: A comprehensive guide to trading methods and applications**. [S.l.]: MIT Press, 1999.

MURPHY, K. P. **Machine Learning: A Probabilistic Perspective**. [S.l.]: The MIT Press: London, UK, 2012.

NAKAGAWA, E. Y. et al. **Revisão Sistemática da Literatura em Engenharia de Software: Teoria e Prática**. [S.l.]: Elsevier Brasil, 2017.

NEELAKANTAN, A. et al. **Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space**. 2015.

OLAH, C. **Understanding lstm networks**. 2018. <http://colah.github.io/posts/2015-08-Understanding-LSTMs>.

PARMEZAN, A. R. S.; BATISTA, G. E. A study of the use of complexity measures in the similarity search process adopted by knn algorithm for time series prediction. In: IEEE. **Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on**. [S.l.], 2015. p. 45–51.

PARMEZAN, A. R. S.; SOUZA, V. M.; BATISTA, G. E. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. **Information Sciences**, v. 484, p. 302 – 337, 2019.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. In: **International conference on machine learning**. [S.l.: s.n.], 2013. p. 1310–1318.

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. [S.l.: s.n.], 2014. p. 1532–1543.

PERCY, C. E.; MEYER, C. F.; LANCASHIRE, I. **Synchronic corpus linguistics: papers from the sixteenth International Conference on English Language Research on Computerized Corpora (ICAME 16)**. [S.l.]: Rodopi, 1996.

PEROZZI, B.; AL-RFOU, R.; SKIENA, S. Deepwalk. **Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining**, ACM, Aug 2014. Disponível em: <http://dx.doi.org/10.1145/2623330.2623732>.

PETERS, M. E. et al. **Deep contextualized word representations**. 2018.

PONTI, M. A.; COSTA, G. B. P. da. Como funciona o deep learning. **arXiv preprint arXiv:1806.07908**, 2018.

QIAO, Y. et al. **Understanding the Behaviors of BERT in Ranking**. 2019.

QIN, Y. et al. A dual-stage attention-based recurrent neural network for time series prediction. **CoRR**, abs/1704.02971, 2017.

RAGHAVAN, U. N.; ALBERT, R.; KUMARA, S. Near linear time algorithm to detect community structures in large-scale networks. **Physical Review E**, American Physical Society (APS), v. 76, n. 3, Sep 2007. ISSN 1550-2376.

REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In: **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing**. [S.l.]: Association for Computational Linguistics, 2019.

RODRIGUES, L. S. et al. Agribusiness time series forecasting using perceptually important events. **2018 XLIV Latin American Computer Conference (CLEI)**, p. 268–277, 2018.

ROKACH, O. M. L. Supervised Learning. In: _____. **Data Mining and Knowledge Discovery Handbook**. Boston, MA: Springer US, 2010. p. 667–685. ISBN 978-0-387-09823-4.

ROSENBLATT, F. The perceptron a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

ROSSI, R. G. **Classificaçao automática de textos por meio de aprendizado de máquina baseado em redes**. Tese (Doutorado) — Universidade de São Paulo, 2015.

ROSSI, R. G.; LOPES, A. de A.; REZENDE, S. O. Optimization and label propagation in bipartite heterogeneous networks to improve transductive classification of texts. **Information Processing & Management**, v. 52, n. 2, p. 217–257, 2016. ISSN 0306-4573. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0306457315000990>.

RUMELHART, D.; MCCLELLAND, J. **Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Foundations. Volume 1**. [S.l.]: MIT Press, 1986. (A Bradford book).

SAHLGREN, M. The distributional hypothesis. **Italian Journal of Disability Studies**, v. 20, p. 33–53, 2008.

SÁNCHEZ, A. Definición e historia de los corpus. **Cumbre: Corpus lingüístico del Español contemporáneo. Madrid: SGEL**, p. 7–24, 1995.

SANH, V. et al. **DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter**. 2020.

SANTOS, A. D. **Análise de Sentimento Multiclasse: Uma abordagem com o uso de Aprendizado de Maquina**. São Carlos, Universidade de São Carlos: Programa de Pós-Graduação em Computação, 2019.

SANTOS, N. d. B.; ROSSI, G. R.; MARCACINI, M. R. Transductive event classification through heterogeneous networks. **WebMedia**, p. 285–292, 2017.

SARDINHA, T. B. Linguística de Corpus: historia e problemática. **DELTA: Documentação de Estudos em Linguística Teórica e Aplicada**, scielo, v. 16, p. 323 – 367, 00 2000. ISSN 0102-4450.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural Networks**, v. 61, p. 85 – 117, 2015. ISSN 0893-6080.

SHIH, S.-Y.; SUN, F.-K.; LEE, H. yi. **Temporal Pattern Attention for Multivariate Time Series Forecasting**. 2018.

Sousa, M. G. et al. Bert for stock market sentiment analysis. In: **2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)**. [S.l.: s.n.], 2019. p. 1597–1601.

SPECIA, L.; NUNES, M. d. G. V. O problema da ambigüidade lexical de sentido na comunicação. In: **Congresso Brasileiro de Computação**. [S.l.: s.n.], 2004.

SUBELJ, L. Label propagation for clustering. In: _____. **Advances in Network Clustering and Blockmodeling**. [S.l.]: John Wiley and Sons, Ltd, 2019. cap. 5, p. 121–150. ISBN 9781119483298.

SUN, C. et al. **How to Fine-Tune BERT for Text Classification?** 2020.

SUNDERMANN, C. et al. Using opinion mining in context-aware recommender systems: A systematic review. **Information**, v. 10, p. 42, 01 2019.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. **Sequence to Sequence Learning with Neural Networks**. 2014.

TAN, P.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining**. [S.l.]: Pearson Addison Wesley, 2006. (Always learning). ISBN 9780321321367.

TEIXEIRA, L. A.; OLIVEIRA, A. L. I. de. A method for automatic stock trading combining technical analysis and nearest neighbor classification. **Expert Systems with Applications**, v. 37, n. 10, p. 6885 – 6890, 2010. ISSN 0957-4174.

VASWANI, A. et al. **Attention Is All You Need**. 2017.

VINCENT, P. A connection between score matching and denoising autoencoders. **Neural Computation**, v. 23, n. 7, p. 1661–1674, 2011.

VINCENT, P. et al. Extracting and composing robust features with denoising autoencoders. In: ACM. **Proceedings of the 25th international conference on Machine learning**. [S.l.], 2008. p. 1096–1103.

_____. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. **Journal of machine learning research**, v. 11, n. Dec, p. 3371–3408, 2010.

VRIGAZOVA, B.; PAVLOVA, T.; BOGDANOVA, B. Wavelet-based Prediction of Stock Market Returns during 2008 Financial Crisis. v. 12, n. 1, 2016.

WALLACE, E. et al. **Do NLP Models Know Numbers? Probing Numeracy in Embeddings**. 2019.

WANG, S. I.; MANNING, C. D. Baselines and bigrams: Simple, good sentiment and topic classification. In: **Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. [S.l.: s.n.], 2012. p. 90–94.

WANG, Y. Stock market forecasting with financial micro-blog based on sentiment and time series analysis. **Journal of Shanghai Jiaotong University (Science)**, v. 22, n. 2, p. 173–179, Apr 2017.

WERBOS, P. J. Generalization of backpropagation with application to a recurrent gas market model. **Neural networks**, Elsevier, v. 1, n. 4, p. 339–356, 1988.

XU, K. et al. Show, attend and tell: Neural image caption generation with visual attention. **CoRR**, abs/1502.03044, 2015.

YOO, P. D.; KIM, M. H.; JAN, T. Machine learning techniques and use of event information for stock market prediction: A survey and evaluation. In: **International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)**. [S.l.: s.n.], 2005. v. 2, p. 835–841.

YOUNG, T. et al. **Recent Trends in Deep Learning Based Natural Language Processing**. 2018.

ZHANG, G. P. Time series forecasting using a hybrid arima and neural network model. **Neurocomputing**, Elsevier, v. 50, p. 159–175, 2003.

ZHANG, L.; WANG, S.; LIU, B. Deep learning for sentiment analysis : A survey. **CoRR**, abs/1801.07883, 2018.

ZHANG, X. et al. AT-LSTM: An attention-based LSTM model for financial time series prediction. **IOP Conference Series: Materials Science and Engineering**, IOP Publishing, v. 569, p. 052037, aug 2019.

ZHOU, D. et al. Learning with local and global consistency. In: THRUN, S.; SAUL, L.; SCHöLKOPF, B. (Ed.). **Advances in Neural Information Processing Systems**. [S.l.]: MIT Press, 2004. v. 16, p. 321–328.

ZHOU, J. et al. **Graph Neural Networks: A Review of Methods and Applications**. 2019.

ZHU, X.; GHAHRAMANI, Z. Learning from labeled and unlabeled data with label propagation. Citeseer, 2002.

ZHU, X.; GHAHRAMANI, Z.; LAFFERTY, J. Semi-supervised learning using gaussian fields and harmonic functions. In: **Proceedings of the Twentieth International Conference on International Conference on Machine Learning**. [S.l.]: AAAI Press, 2003. (ICML'03), p. 912–919. ISBN 1577351894.