Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Classificação de títulos de notícias do mercado financeiro brasileiro utilizando modelos de aprendizado de máquina

Guilherme Coelho Minervino

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientadora
Prof.a Dr.a Roberta Barbosa Oliveira

Brasília
2023

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Classificação de títulos de notícias do mercado financeiro brasileiro utilizando modelos de aprendizado de máquina

Guilherme Coelho Minervino

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof.a Dr.a Roberta Barbosa Oliveira (Orientadora)
CIC/UnB

Prof. Dr. Vinicius Ruela Pereira Borges
CIC/UnB

Prof. Dr. Thiago Paulo Faleiros
CIC/UnB

Prof. Dr. João Luiz Azevedo de Carvalho
Coordenador do Curso de Engenharia da Computação

Brasília, 20 de julho de 2023

Dedicatória

Dedico este trabalho a todos aqueles que se interessam pelo maravilhoso mundo dos dados, mais especificamente, na área de aprendizado de máquina e processamento de linguagem natural. Dedico também aos interessados em investimentos e estudar sobre o mercado financeiro.

Agradecimentos

Agradeço à minha mãe, Kárin, e ao meu pai, Marcus, que me apoiaram e investiram para que eu tenha uma vida profissional e emocional saudável. Agradeço à minha psicanalista, Liliam, por nosso maravilhoso trabalho analítico e ótimas conquistas! E também aos colegas: Vieira, Mayara, Rafael e Chaves, por todos os aprendizados e companheirismo ao longo do curso. Um agradecimento especial à Profa Dra Roberta Barbosa Oliveira pela ótima orientação, explicações e paciência. Também agradeço ao Prof Dr Vinicius Ruela Pereira Borges por todas as conversas, mentorias e ensino. Por fim, mas não menos importante, gostaria de agradecer à Empresa Júnior de Computação - CJR, por me proporcionar conexões, experiências e aprendizados que levarei por toda vida.

Resumo

O mercado financeiro é extremamente importante para o desenvolvimento saudável da economia de um país. Com bilhões de reais movimentados diariamente no Brasil e sob influência de diversos fatores externos como política, economia, desastres naturais e doenças, torna-se fundamental ser discutido na literatura como analisar e prever esse mercado. Dado esse domínio, este trabalho busca auxiliar os *stakeholders* deste mercado ao contribuir para a recuperação de informações de notícias e a predição do preço de ações. Para isso, explora a classificação de notícias do mercado financeiro brasileiro nas classes Petrobras, Vale, Itaú ou Outros; e a predição do movimento do preço da ação da Petrobras (PETR4). Este trabalho aplica conceitos de *Natural Language Processing* e *Machine Learning* para lidar com as características não-estruturadas e ruidosas dos dados textuais gerados pelas notícias. Para executar essas duas tarefas, um método é proposto com o objetivo de organizar as etapas de processamento dos dados e das predições. Para a realização dos experimentos deste trabalho, foram considerados modelos de aprendizado tradicional e de *deep learning*. Os experimentos utilizam os modelos de aprendizado tradicional *K-Nearest Neighbors* (KNN), *Support-Vector Machine* (SVM), *Naive Bayes* (NB) e *Logistic Regression* (LR), e os modelos de *deep learning Long Short-Term Memory* (LSTM) e *Bi-directional Long Short-Term Memory* (Bi-LSTM). A base de dados deste trabalho foi obtida de um repositório público contendo notícias de diferentes portais brasileiros. Os resultados experimentais das tarefas foram avaliados utilizando as métricas acurácia e *F1-Score*. Além disso, foi aplicada a técnica *holdout* com *splits* de treino, validação e teste para o treinamento e avaliação dos modelos preditivos. A classificação de notícias obteve resultados promissores, com o melhor modelo sendo o LSTM com a representação *word2vec*, com acurácia e *F1-Score* de 83.07% e 81.19%, respectivamente. Para a tarefa de predição do movimento da ação Petrobras, os resultados não foram satisfatórios, com melhor modelo sendo SVM com a representação *Term Frequency - Inverse Document Frequency*, atingindo a acurácia de 50.68% e *F1-Score* de 34.01%.

Palavras-chave: Processamento de linguagem natural, Aprendizado de máquina, *Deep learning*, Classificação de texto, Predição no mercado financeiro

Abstract

The financial market is essential for the healthy development of a country's economy. With billions of *reais* being moved daily in Brazil and under the influence of several external factors such as politics, economics, natural disasters, and diseases, it becomes crucial to be discussed in the literature. Given this domain, this work helps stock market stakeholders with information recovery of news and stock market share prediction. Therefore, the classification of financial market news in the classes Petrobras, Vale, Itaú or Others, and the stock price forecasting of Petrobras (PETR4) are explored. Hence, Natural Language Processing and Machine learning are used to deal with the unstructured and noisy characteristics of the textual data generated by the news. Here, a method is proposed to perform these tasks, with the goal of organizing data processing and predictions of this study. Considering the experiments of this study, traditional machine learning and deep learning are applied. K-Nearest Neighbors (KNN), Support-Vector Machine (SVM), Naive Bayes (NB), and Logistic Regression (LR) compose the traditional machine learning models and Long Short-Term Memory (LSTM) and Bi-directional Long Short-Term Memory (Bi-LSTM) represent the deep learning models. The database was obtained from a public repository containing news from different Brazilian portals. The experiment results of the tasks executed by this work were evaluated by accuracy and F1-Score metrics. Besides that, the holdout technique was applied, with train, validation, and test splits to train and evaluate the predictive models. The task of news classification showed promising results, with the best model being LSTM combined with word2vec, scoring 83.07% and 81.19% for accuracy and F1-Score metrics, respectively. The task of PETR4 stock price forecasting did not show interesting results, since the best model using SVM with Term Frequency - Inverse Document Frequency achieved an accuracy of 50.68% and F1-Score of 34.01%.

Keywords: Natural Language Processing, Machine learning, Deep learning, Text classification, Stock market forecasting

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AF Análise fundamental.

ANN Artificial Neural Networks.

AT Análise técnica.

B3 Brasil, Bolsa e Balcão.

BERT *Bidirectional Encoder Representations for Transformers.*

Bi-GRU *Bi-directional Gated Recurrent Unit.*

Bi-LSTM *Bi-directional Long Short-Term Memory.*

BoW *Bag of Words.*

CNN *Convolutional Neural Network.*

DJI *Dow Jones Index.*

GRU *Gated Recurrent Unit.*

KNN *K-Nearest Neighbors.*

KOSPI *Korea Stock Price Index.*

LR Logistic Regression.

LSTM *Long Short-Term Memory.*

ML *Machine learning.*

MLP *Multi-Layer Perceptron.*

MSE *Mean Squared Error.*

NB *Naive Bayes*.

NLP *Natural Language Processing*.

NLTK **Natural Language Toolkit**.

REGEX Expressões Regulares, do Inglês *Regular Expressions*.

RMSE *Root Mean Squared Error*.

RNN Rede Neural Recorrente, do Inglês *Recurrent Neural Network*.

SVM *Support-Vector Machine*.

TF-IDF *Term Frequency - Inverse Document Frequency*.

Capítulo 1

Introdução

Este capítulo apresenta as motivações para realização deste trabalho, ao contextualizar o problema a ser tratado, objetivos gerais e específicos, que serve de guia para os experimentos e análises dos resultados, e a estrutura do documento, que descreve como o trabalho é organizado.

1.1 Contextualização

O mercado financeiro é fundamental para a economia de um país. É por esse tipo de mercado que as empresas recebem investimento, e em troca, os investidores recebem uma parte dela, sendo chamada de ação. Dessa forma, tanto a empresa quando o investidor podem crescer juntos financeiramente e realizar seus sonhos. Ademais, o número de brasileiros que investem está aumentando a cada ano, com 4,2 milhões inscritos na B3[1] em 2022; um aumento de 15% de CPFs em comparação ao ano anterior[2]. Movimentando bilhões de reais diariamente, esse mercado provoca interesse não só de investidores, mas de pesquisadores e analistas financeiros. Porém, devido à crescente quantidade de dados textuais de notícias geradas atualmente, torna-se difícil a recuperação da informação desejada por esses grupos, bem como a predição da movimentações das ações em um determinado dia. Muitos trabalhos têm sido feitos na literatura para contribuir na resolução desses problemas.

Com o objetivo de tratar do problema da recuperação da informação de notícias, trabalhos exploram a tarefa de classificação de textos (organizar textos em classes) ao classificar (ou categorizar) esses textos em classes como negócios, economia, esportes, e

[1] https://www.b3.com.br/pt_br/

[2] https://www.cnnbrasil.com.br/economia/numero-de-investidores-na-bolsa-cresce-15-em-2022-apostando-na-diversificacao

entretenimento [3] [4] [5]. Com isso, as notícias ficam organizadas de forma mais eficiente e automatizada, facilitando a busca por informação.

Em relação à tarefa de predição do valor de uma ação, a literatura aplica a análise técnica e fundamental, análises extensivamente aplicadas para prever a movimentação do mercado financeiro [6]. A análise técnica utiliza de dados históricos quantitativos do passado e presente da ação para se prever o seu valor futuro. Isso é feito ao encontrar padrões e picos que se repetem ao se analisar a variação do preço em um intervalo de tempo. Existem trabalhos propostos na literatura que aplicam essa análise para a predição dos preços [7] [8]. A análise fundamental estima o valor da ação se baseando em fatores como notícias, mídias sociais, política, gerenciamento interno e externo da empresa e variáveis macro-econômicas, fatores considerados mais qualitativos que os fatores levados em consideração para fazer a análise técnica. Essa análise é explorada na literatura por meio de classificação (ou categorização) de textos, ao classificar sentimentos de notícias e *tweets* [9] [10] [11]. Além disso, também há estudos que combinam as duas análises para fazer a predição dos preços, aproveitando dos benefícios das análises dos dados históricos (resultado da análise técnica) e também dos dados textuais de notícias, *tweets* e fóruns (resultado da análise fundamental) [1] [12]. Portanto, observa-se vários trabalhos na literatura que contribuem para a predição de preços do mercado financeiro aplicando conhecimentos de *Machine learning (ML)*, otimizando os resultados dos investidores do mercado financeiro que decidam considerar essas ferramentas no seu trabalho.

Para realizar as classificações de textos, os trabalhos apresentados na literatura aplicam de forma extensiva *ML* ao aplicar modelos de *deep learning* e modelos tradicionais de aprendizado de máquina (que não são considerados de *deep learning*) [3] [13]. Entre os modelos de aprendizado tradicional mais considerados na literatura, estão *K-Nearest Neighbors (KNN)*, *Support-Vector Machine (SVM)*, *Naive Bayes (NB)*, *Logistic Regression (LR)* e *Multi-Layer Perceptron (MLP)* [11] [13]. Já em relação aos modelos de *deep learning*, *Long Short-Term Memory (LSTM)*, *Bi-directional Long Short-Term Memory (Bi-LSTM)* e *Convolutional Neural Network (CNN)* estão presentes [1] [4].

Em relação aos trabalhos citados que utilizam análise fundamental, tipo de análise que este trabalho aprofunda, dados textuais são muito utilizados. Para que os modelos preditivos compreendam esses dados, os trabalhos aplicam conhecimentos de *Natural Language Processing (NLP)* [11] [12]. Entre esses conhecimentos, está a normalizações de textos e extração de características. A normalização de textos é importante para transformar os dados textuais em um formato mais padronizado para que possa ser processado com mais facilidade em outras análises. Entre as técnicas que os trabalhos utilizam estão a transformação de texto, filtros com *Expressões Regulares, do Inglês Regular Expressions (REGEX)*, tokenização e remoção de *stopwords*. Já ao considerar as abordagens para

extração de características, as representações numéricas *Bag of Words (BoW)* e *Word Embeddings* também são bastante aplicadas [5] [14]. As representações *BoW* mais presentes nos experimentos apresentados na literatura contemplam o tipo Contagem e *Term Frequency - Inverse Document Frequency (TF-IDF)*. Dada à abordagem *Word Embeddings*, as técnicas *word2vec* e *GloVe* são empregadas.

Apesar das contribuições da literatura na área de classificação de notícias, não foi encontrada na revisão deste trabalho a classificação dessas notícias considerando empresas como classes a serem preditas (por exemplo, predizer se um título de notícia se refere à Petrobras, Vale ou Itaú), uma das tarefas que o presente trabalho irá explorar. Ademais, em relação a tarefa de predição do preço de ações do mercado financeiro, observou-se várias contribuições da literatura, mas foram percebidos poucos trabalhos de predição do mercado financeiro brasileiro, principalmente considerando a análise fundamental [6]. Além disso, percebeu-se na literatura várias contribuições na classificação de notícias considerando mercados financeiros estrangeiros. Porém, o mercado financeiro brasileiro foi pouco explorado, ainda mais comparando o desempenho entre modelos tradicionais e de *deep learning*. Portanto, este trabalho tem o intuito de auxiliar na recuperação de informações de notícias e na predição da movimentação de ações do mercado financeiro brasileiro. A base de dados utilizada neste trabalho é composta por dados quantitativos (dados históricos) e qualitativos (títulos de notícias), ambos extraídos de fontes públicas da internet.

1.2 Objetivos Gerais e Específicos

Este trabalho considera dois objetivos gerais: (1) classificar títulos de notícias do mercado financeiro brasileiro nas classes Petrobras, Vale, Itaú ou Outros; e (2) classificar (ou prever), por meio da análise fundamental, se o movimento do preço da ação da Petrobras (PETR4) foi de descida, permaneceu constante, ou subiu, também utilizando títulos de notícias. Para que os objetivos gerais citados sejam cumpridos, os seguintes objetivos específicos foram definidos:

- Avaliar o desempenho das diferentes abordagens de extração de características.
- Comparar o desempenho entre modelos de aprendizado de máquina tradicionais e de *deep learning*;

A partir dos objetivos gerais e específicos, espera-se, por meio dos experimentos e conclusões geradas, auxiliar os *stakeholders* do mercado financeiro brasileiro (investidores, analistas e pesquisadores) na recuperação e organização de informações relevantes de notícias bem como contribuir para a tomada de decisão dos seus investimentos.

1.3 Estrutura do documento

A estrutura deste documento é organizada da seguinte forma: o Capítulo 2 aborda a fundamentação teórica, que expõe a teoria estudada para realizar os experimentos deste trabalho; o Capítulo 3 apresenta a revisão de literatura, contendo os principais trabalhos que serviram de inspiração e embasamento para os experimentos; o Capítulo 4 descreve as tarefas, bem como a metodologia seguida para suas realizações; o Capítulo 5 expõe os resultados experimentais dessas tarefas; o Capítulo 6 aborda as discussões e análises dos resultados dos experimentos; e por fim, o Capítulo 7 apresenta as conclusões finais deste trabalho, além de propostas de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta os principais conceitos que foram utilizados para a realização deste trabalho, que busca classificar títulos de notícias nas classes Petrobras, Vale, Itaú ou Outros; além de realizar uma análise fundamental por meio de títulos de notícias para prever se o preço da ação da Petrobras (PETR4) irá descer, permanecer constante ou subir.

2.1 Mercado de ações e suas análises de predição

O mercado de ações é o lugar em que a sociedade pode recorrer caso queira ser sócia de parte de uma empresa, fazendo com que tanto essa quanto o investidor se beneficiem e cresçam juntos. Porém, para fazer isso, é necessário estar inscrito em uma bolsa de valores. No caso do Brasil, é necessário estar na B3. Devido à grande quantidade de dados textuais gerados diariamente, torna-se difícil organizá-los e tomar decisões que proporcionem retorno financeiro aos investidores. Para auxiliar nesse processo, observa-se na literatura trabalhos que utilizam conceitos de *NLP* e *ML*. Esse auxílio é feito ao classificar esses dados ou usá-los para análises que buscam prever o movimento do preço de ações. Com o objetivo de realizar a predição desse movimento, observa-se na literatura que as análises técnica e fundamental são as mais utilizadas [1] [6].

A análise técnica busca prever o valor de uma ação se baseando no comportamento do seu preço no presente e no passado, com o objetivo de identificar padrões que se repetem, bem como possíveis picos de variação em seu valor. Para investigar esse comportamento, é necessário coletar os dados históricos do preço da ação em um determinado período. Um exemplo considerando dados históricos do índice brasileiro Ibovespa, disponível no site do Yahoo Finance[1] é ilustrado na Figura 2.1, sendo o eixo Y dado em pontos e o eixo

[1] https://finance.yahoo.com/

X, a data da pontuação feita. No Brasil, costuma-se acompanhar esse índice por meio da pontuação em vez do seu preço, mas a análise técnica pode ser aplicada da mesma forma.

Figura 2.1: Exemplo de uma fonte que possa ser utilizada para análise técnica (fonte: *Yahoo Finance*).

Já a análise fundamental ou fundamentalista, procura prever o valor da ação se baseando em fatores como notícias, mídias sociais, política, além de documentos e qualidade da gestão da empresa. Devido ao fato de muitas vezes serem dados não estruturados e de diversas fontes, torna-se difícil haver um único tipo de abordagem para analisar e predizer o futuro do valor de uma ação. Isso se confirma ao se observar na literatura artigos utilizando de diferentes fontes de dados textuais para realizar essa predição [10] [11] [12]. Além disso, está análise possui a capacidade de identificar qualidades ou defeitos de uma empresa, que muitas vezes não estão refletidos no seu preço declarado na bolsa. Um exemplo recente disso é o caso da varejista Americanas[2]. O valor de sua ação só despencou após o mercado financeiro perceber que a empresa fraudou seus documentos contábeis. Porém, esse potencial de queda da ação esteve presente desde o momento em que a contabilidade da empresa começou a ser fraudada e corrompida, mas estava escondido em seus documentos internos, em vez de informados nos dados históricos. A Figura 2.2, retirada do portal *Suno Research*[3], ilustra um exemplo de dado textual em que pode ser aplicada a análise fundamental.

[2]https://www.moneytimes.com.br/a-maior-fraude-da-historia-corporativa-do-brasil-o-caso-americanas-amer3/

[3]https://www.suno.com.br/noticias/ibovespa-fechamento-260523-110-mil-pontos

Figura 2.2: Exemplo de uma fonte textual que pode ser utilizada para análise fundamental (fonte: Portal *Suno Research*).

2.2 Processamento de Linguagem Natural

Devido ao fato das máquinas entenderem apenas números, foi necessário encontrar uma forma dos textos também serem entendidos por elas. A área de *NLP* é responsável por esta tarefa [15]. A solução deste problema permite que várias outras tarefas possam ser executadas, como: classificação de textos, análise de sentimentos, recognição de fala e tradução. Em *NLP*, cada amostra de dados textuais pode ser considerada um documento. Um documento é formado por *tokens* (ou palavras); e um *corpus* é formado por uma coleção de documentos. Neste trabalho, cada título de notícia é considerado um documento. Para esses documentos serem entendidos pelos modelos preditivos, é necessário aplicar um pré-processamento utilizando *NLP*.

2.3 Pré-processamento

Entende-se por pré-processamento todas as etapas necessárias para transformar os dados brutos (recém extraídos das fontes de dados) em um formato apropriado para tarefas de mineração de dados, como a utilização de modelos de aprendizado de máquina [16]. Este trabalho realiza pré-processamentos de dados textuais, constituindo de duas etapas: normalização de textos e extração das suas características.

2.3.1 Normalização de textos

A normalização consiste em converter textos para uma forma mais conveniente e padronizada para que possam ser processados mais facilmente em etapas futuras, como análises estatísticas ou extrações de características [15]. A normalização costuma se dar pelas seguintes etapas: transformação do texto para uma única tipografia (*lowercase* ou *upper-*

case), aplicação de expressões regulares, tokenização, remoção de *stopwords* e stemização ou lematização [17].

Transformação de textos

Esta etapa exerce importante função na padronização dos textos. Caso não existisse essa etapa, termos que possuem sentido equivalente, como "lençol" e "Lençol", seriam tratados de forma diferente por não possuírem a mesma tipografia. Além disso, também é responsabilidade desta etapa a conversão de palavras em símbolos e *emojis*, ou vice-versa. Por exemplo, o *emoji* ":)" pode ser convertido para a palavra "feliz".

Expressões regulares

Resultado do trabalho do matemático norte-armericano Stephen Cole Kleene [18], as *Expressões Regulares, do Inglês Regular Expressions (REGEX)* são muito utilizadas por sua rica sintaxe, que possibilita criações de infinitas expressões que servem para diversos propósitos diferentes. Tarefas como verificar se o preenchimento do CPF em um formulário de cadastro está no formato correto e encontrar datas em formato português de uma notícia podem ser realizadas de forma rápida e elegante. Essas expressões também podem ser aplicadas para remover acentos, números e caracteres especiais do *corpus*, já que não agregam informação de forma intensa ao sentido texto, sendo considerados ruídos. Com essas remoções feitas, o desempenho de etapas como extração de características e predições costuma apresentar resultados mais satisfatórios [15].

Tokenização

Existe também o processo de tokenização, cuja função é encontrar as menores partes divisíveis que contenham sentido em um documento (*token*) e guardá-los para que cada um vire uma característica do texto. Em outras palavras, *token* é definido como o menor conjunto de caracteres que possuem sentido. Muitas vezes, os *tokens* podem ser encontrados entre dois caracteres do tipo espaço ou pontuações, porém, existem casos que isso não é verdade. Por exemplo, ao considerar o vocabulário da língua inglesa, a sequência de caracteres *"aren't"*, pode ser dividida em *"aren"* e *"t"*, mas estas não possuem sentido quando separadas, mas sim juntas.

Remoção de *stopwords*

As *stopwords* são palavras que, assim como acentos, números e caracteres especiais, são muitas vezes consideradas ruído. Classes de palavras como preposições, conjunções e arti-

gos costumam se enquadrar nessa categoria. Portanto, também costumam ser removidas do *corpus*.

Stemização e Lematização

Outros dois métodos para normalização de textos muito utilizados na literatura são stemização e lematização. O primeiro método consiste em remover sufixos de uma palavra, ou seja, a palavra resultante nem sempre existe no vocabulário da língua. O segundo busca reduzir uma palavra em sua raiz de origem, necessitando de um dicionário para realizar essa redução. Portanto, no caso da lematização, a palavra resultante sempre existe no vocabulário do idioma em questão [17]. Por exemplo, ao aplicar stemização em "cantei", "cantou", "cantaram" e "cantando", o termo resultante é "cant". Já ao aplicar lematização, o termo resultante é "cantar".

2.3.2 Extração de características

Após a etapa de normalização de textos, os dados textuais estão mais preparados para poderem ter suas características extraídas. Para isso, as abordagens de representações numéricas desses textos são bastante utilizadas na literatura [19]. A representação numérica consiste em transformar os dados textuais em um domínio que possa ser entendido pelas máquinas de aprendizagem, isto é, números. Ela pode ser aplicada levando em consideração apenas a frequência das palavras em um documento ou *corpus* ou aplicada levando em consideração seu contexto. Dessa forma, essa forma de extração de características busca extrair a maior quantidade possível de características que possam ser processadas pelos modelos preditivos. Neste trabalho, as representações *BoW* e *Word embedding* são empregadas.

Bag of Words

A representação *BoW* leva em consideração apenas a frequência de cada palavra em um documento ou *corpus*, desconsiderando posição e contexto [17]. Dois tipos de *BoW* são amplamente aplicados na literatura: tipo Contagem e tipo *TF-IDF* [17].

No caso do *BoW* do tipo Contagem, é levando em consideração apenas a frequência das palavras em um único documento. O cálculo de $\mathrm{TF}_{t,d}$ (*term frequency*) é feito por meio da Equação 2.1, em que os parâmetros t e d são a palavra e o documento que esta palavra está contida, respectivamente. A função *count* retorna a quantidade de vezes que a palavra foi mencionada nesse documento.

$$\mathrm{TF}_{t,d} = count(t, d) \tag{2.1}$$

O tipo *TF-IDF* também leva em consideração a frequência das palavras em um documento, mas considera o inverso da frequência dessas palavras em todo o *corpus* (todos os documentos). Esse tipo de representação é calculado se baseando na Equação 2.2, em que $\text{TF-IDF}_{t,d}$ (*Term Frequency - Inverse Document Frequency*) é resultado da multiplicação de $\text{TF}_{t,d}$ (frequência de uma dada palavra no documento) com IDF_t (inverso da frequência do termo considerando todos os documentos). Ou seja, quanto mais frequente uma palavra é em todos os documentos do *corpus*, menor o valor de IDF_t), e consequentemente, menor o valor do TF-IDF.

$$\text{TF-IDF}_{t,d} = \text{TF}_{t,d} * \text{IDF}_t \tag{2.2}$$

Por exemplo. ao considerar dois documentos normalizados textualmente d_0 e d_1 (retirados de um dos *corpus* deste trabalho):

- d_0 - *petrobras aprova cessao participacao campos petroleo*;
- d_1 - *mercado comeca precificar fortalecimento alckmin*;

sendo os seus *BoW*s do tipo contagem representados na Tabela 2.1, e os do tipo *TF-IDF* na Tabela 2.2 (algumas palavras não foram mostradas para melhorar a visualização das tabelas). Ao observar as palavras "petrobras" e "petroleo", percebe-se que ambas têm valor 1 no *BoW* de d_0, mas no *TF-IDF*, o valor é de 0.201 e 0.372, respectivamente; indicando que apesar desta palavra ocorrer nesse documento, ela está muito frequente nos demais documentos do *corpus*, sendo penalizada. Um dos problemas da representação *BoW* é que ela gera vetores esparsos e longos por conta do seu tamanho ser diretamente proporcional ao vocabulário do *corpus* [17], tornando seu processamento pelos modelos preditivos mais lento.

Tabela 2.1: Dois exemplos de BoW do tipo Contagem.

	comeca	fortalecimento	mercado	participacao	petrobras	petroleo
d_0	0	0	0	1	1	1
d_1	1	1	1	0	0	0

Tabela 2.2: Dois exemplos de *BoW* do tipo *TF-IDF*.

	comeca	fortalecimento	mercado	participacao	petrobras	petroleo
d_0	0.000	0.000	0.000	0.430	0.201	0.372
d_1	0.406	0.502	0.260	0.000	0.000	0.000

Word Embeddings

Esta representação soluciona o problema dos vetores esparsos do *BoW*, comentado na seção anterior, ao gerar vetores densos. Para isso, cada documento é transformado em uma representação vetorial, cuja cada posição é um índice que representa uma palavra contida no documento. Além disso, cada índice aponta para seu vetor de incorporação de palavras correspondente, também chamado de *Word Embedding*, armazenando o contexto da palavra. Esse contexto é criado a partir do treinamento de uma *Artificial Neural Networks (ANN)* no *corpus* que contém essas palavras. Palavras que possuem um contexto parecido possuem vetores *embeddings* parecidos. Por exemplo, ao final do treinamento de uma rede desse tipo, palavras como "petróleo", "petrobras" e "gás" teriam *embeddings* com valores parecidos, pois essas palavras frequentemente aparecem próximas uma das outras em textos, ou seja, normalmente estão no mesmo contexto, no caso deste exemplo, no contexto da produção de petróleo.

2.4 Aprendizado de Máquina

Pode-se concluir que uma máquina aprende a partir de uma experiência E em uma tarefa T se sua performance P melhora com a experiência E [20]. Existem vários tipos de aprendizado de máquina (por exemplo, supervisionado, não supervisionado, semi-supervisionado e por reforço). Nos experimentos, será utilizado apenas o aprendizado de máquina supervisionado, uma vez que esse tipo de aprendizado é realizado a partir de experiências de dados passados e rotulados; característica observada nos dados deste trabalho. Considerando que os dados textuais estão em sua representação numérica, é possível passá-los aos modelos de aprendizado de máquina para as tarefas de classificação.

O surgimento das *ANNs* modernizou a forma como os modelos de aprendizado de máquina são treinados e postos em prática. Tendo como inspiração o cérebro humano, as *ANNs* são compostas por camadas de neurônios artificiais. Com o aumento de complexidade da conexão das suas redes, surgiu o conceito de *deep learning* para as redes neurais. Um dos tipos de modelos de *deep learning* são as *RNNs*, que são aplicadas neste trabalho pelo seu uso abundante na literatura na área de classificação de títulos de notícias [5] [1]. Isso se deve ao fato desta rede possuir a característica de permitir sequencias de entradas de tamanhos diferentes, comportamento visto nos dados textuais. Portanto, são abordados nesta seção modelos de aprendizado de máquina tradicional, como *KNN*, *SVM*, *NB* e *LR*, e também modelos de *deep learning*, como *LSTM* e *Bi-LSTM*.

2.4.1 *K-Nearest Neighbors*

Proposto pelos autores Fix e Hodges [21], o *KNN* é um modelo não paramétrico, pois este modelo não possui parâmetros que têm seu valor atualizado durante o treinamento. Em vez disso, é utilizado a distância entre a instância a ser predita e as demais amostras, sendo K o número de amostras cuja distância será levada em consideração para classificar a instância. Por exemplo, considerando K igual a três, apenas as três amostras mais próximas dessa instância terão influência na predição de seu rótulo, sendo a distância determinada pelos valores dos atributos (características) de cada amostra. A decisão da categoria é por voto majoritário. Em caso de empate (mesma quantidade de amostras para todas as categorias), a escolha é aleatória. Por isso, é recomendado que K seja sempre ímpar.

Devido ao fato dos documentos possuírem tamanhos diferentes, seus vetores numéricos correspondentes também possuem tamanhos diferentes, impossibilitando a utilização da distância euclidiana, que precisa de vetores de tamanhos iguais. Portanto, a distância cosseno é empregada neste trabalho para os cálculos de distância. Por exemplo, ao considerar dois documentos representados pelos vetores $\vec{d_1}$ (Expressão 2.3) e $\vec{d_2}$ (Expressão 2.4), e seus *tokens* representados por t, sua similaridade cosseno (s) e distância cosseno ($dist$) é calculada por meio das Equações 2.5 e 2.6, respectivamente.

$$\vec{d_1} = (t_{11}, t_{12}, t_{13}, ..., t_{1n}) \tag{2.3}$$

$$\vec{d_2} = (t_{21}, t_{22}, t_{23}, ..., t_{2p}) \tag{2.4}$$

$$s(\vec{d_1}, \vec{d_2}) = cos\theta = \frac{\vec{d_1}.\vec{d_2}}{||d_1||.||d_2||} \tag{2.5}$$

$$dist(\vec{d_1}, \vec{d_2}) = 1 - s(\vec{d_1}, \vec{d_2}) \tag{2.6}$$

sendo possível encontrar o valor de s por meio de duas formas: (1) calculando o cosseno do ângulo entre esses vetores ou (2) fazendo o produto escalar entre os vetores, dividido pela multiplicação dos seus módulos (tamanhos).

2.4.2 *Support Vector Machine*

Proposto por Cortes e Vapnik [22], as *SVMs* implementam um classificador do tipo *Soft Margin Classifier*, que possuem a propriedade de lidar bem com *outliers* e possíveis erros de classificação. Para fazer a separação, essa máquina cria um *threshold* cuja dimensão é

$D-1$, sendo D a dimensão dos dados a serem preditos. Por exemplo, se os dados forem bi dimensionais, o *threshold* será unidimensional, ou seja, uma reta. A partir do treinamento, esse *threshold* é parametrizado para se ajustar aos dados e permitir a diferenciação das classes do *dataset* rotulado.

2.4.3 *Naive Bayes*

Este modelo faz parte do grupo de modelos probabilísticos e utiliza o conceito do Teorema de Bayes como base para seu funcionamento [23]. Por ser uma generalização de um teorema muito complexo, o nome *Naive Bayes* foi dado, devido a ser um modelo "ingênuo" do original. Isso se deve ao fato do modelo considerar independentes todas as variáveis das amostras a serem preditas em uma dada classe.

2.4.4 *Logistic Regression*

Apesar de possuir um algoritmo de natureza de regressão, o modelo de *LR* é utilizado para classificação, também considerando o conceito de probabilidade para fazer as predições [24] do tipo bi-classe. Dado uma variável dependente Y (categoria a ser predita) e as variáveis independentes x (atributos da amostra), o modelo indica a probabilidade da variável Y assumir uma das categorias. Para ser possível aplicar este modelo em tarefas multi-classe, sua função de perda precisa ser atualizada para ser do tipo *cross-entropy* e sua distribuição de probabilidade atualizada para o tipo multinomial. Ambas atualizações são implementadas pela biblioteca **Sci-kit learn**[4], que é utilizada no presente trabalho.

2.4.5 *Long Short-term Memory*

Proposto por Hochreiter e Schmidhuber [25], as *LSTMs* são um tipo de *RNN* que resolvem o problema de explosão de gradiente, que dificulta a capacidade do modelo de ter uma memória robusta para armazenar informações de longo e curto prazo que passam pela rede. A Figura 2.3 descreve a arquitetura de uma célula *LSTM*. Essa célula possui três portões que recebe as entradas da célula em seu estado anterior (c_{t-1} e h_{t-1}) e que, se baseando na entrada atual x_t e nos portões, decide o que será adicionado ou removido para formar o estado atual, composto pelas saídas c_t e h_t. No caso deste trabalho, cada valor de x_t é um *embedding* de uma palavra contida em um documento. Para que o trabalho dos portões seja possível, as funções *sigmoid* (σ) e *tanh* são aplicadas, conforme mostrado no esquemático da célula *LSTM*.

[4]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

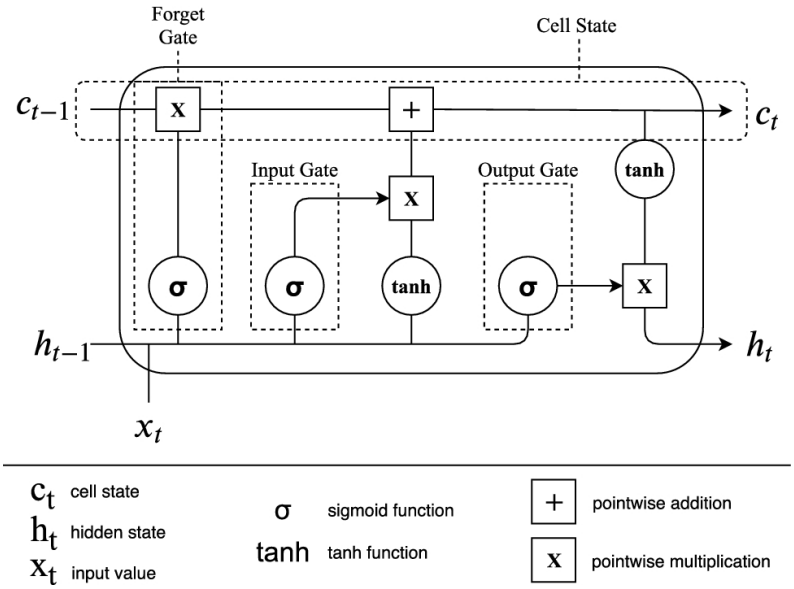


Figura 2.3: Esquemático de uma célula *LSTM* (fonte: Renato *et al.* [1]).

O fato das *LSTMs* armazenarem o contexto dos dados textuais apenas da esquerda para a direita motivou Shu *et al.* [26] a proporem uma arquitetura composta por duas redes de *LSTMs*, as *Bi-LSTMs*, em que uma rede irá analisar o contexto do texto da esquerda para a direita e a outra, o contrário.

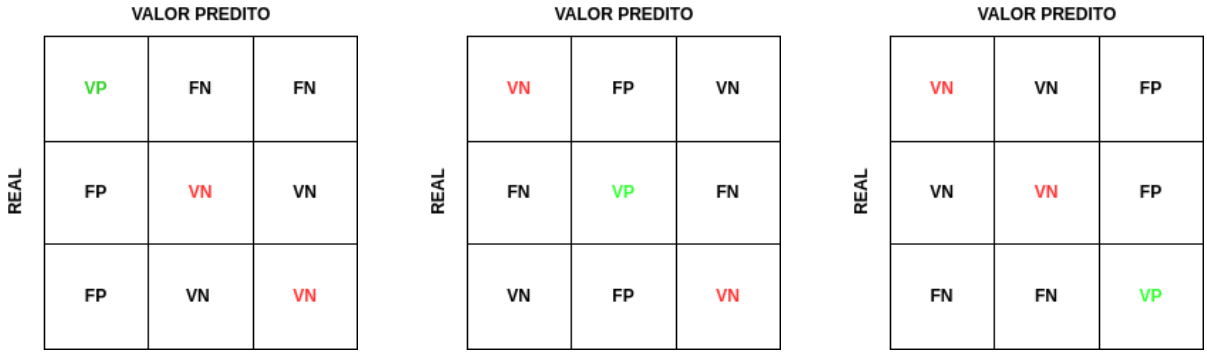
2.5 Métricas de desempenho de classificação

Para avaliar o desempenho de modelos preditivos em tarefas de classificação, são consideradas várias métricas, sendo o cálculo dessas considerando resultados de uma matriz de confusão. As métricas aplicadas neste trabalho são Acurácia, Precisão, Revocação e *F1-Score*.

Matriz de confusão

A matriz de confusão é uma tabela que promove uma fácil visualização do desempenho de modelos de classificação. Ou seja, cada predição é considerada um:

- Verdadeiro Positivo (VP): se a amostra é classificada como pertencendo à classe Positiva e a classe, de fato, é Positiva;
- Falso Positivo (FP): se a amostra é classificada como pertencendo à classe Positiva e a amostra, na verdade, pertence à outra;



(a) Petrobras (Classe 1) (b) Vale (Classe 2) (c) Itaú (Classe 3)

Figura 2.4: Matriz de Confusão para classificação de três classes (fonte: Sousa [2]).

- Verdadeiro Negativo (VN): se a amostra é classificada como pertencendo à classe Negativa e a classe, de fato, é Negativa;
- Falso Negativo (FN): se a amostra é classificada como pertencendo à classe Negativa e a amostra, na verdade, pertence à outra.

A quantidade e o formato das matrizes de confusão depende da quantidade de classes (n) da tarefa, sendo gerada n matrizes com n linhas e n colunas cada. Por exemplo, ao considerar uma tarefa para classificar notícias como sendo pertencentes à três classes (Petrobras, Vale e Itau), as matrizes resultantes estão ilustradas na Figura 2.4, em que as Figuras 2.4a, 2.4b e 2.4c, representam as matrizes de confusão considerando como Positivo a classe Petrobras, Vale e Itaú, respectivamente. Ao somar essas três matrizes, obtêm-se uma matriz final, em que a diagonal principal contém os VPs das classes Petrobras, Vale e Itaú, respectivamente.

Acurácia

Essa métrica de avaliação transparece a relação de predições corretas e o total de predições. Sua equação é dada por:

$$Acurácia = \frac{VP + VN}{VP + FP + VN + FN} \tag{2.7}$$

Normalmente, esta métrica não é utilizada sozinha na literatura devido ao fato de muitas vezes esconder o real desempenho de um modelo preditivo. Por exemplo, ao criar um modelo para classificar e-mails em spam ou não spam que classifica todos os e-mails como sendo spam, se 90% dos dados de teste forem spam, esse modelo vai ter 90% de acurácia, sendo que ele não classificou certo nenhuma amostra considerada não spam. Portanto, outras métricas são apresentadas para serem utilizadas em conjunto e sanar essa deficiência.

Precisão

A precisão mede a relação entre as predições positivas verdadeiras e o total de predições positivas. Portanto, ela transparece o quão bem o modelo foi em classificar predições verdadeiras. Quanto maior o valor da precisão, menor a quantidade de falsos positivos. Sua equação é descrita por:

$$Precisão = \frac{VP}{VP + FP} \tag{2.8}$$

Revocação

Mede a relação entre as predições positivas verdadeiras e o total de positivos. Portanto, transparece a relação entre a quantidade de positivos verdadeiros em relação a todos os positivos. Quanto maior o valor da revocação, menor a quantidade de falsos negativos. Seu cálculo é regido pela equação:

$$Revocação = \frac{VP}{VP + FN} \tag{2.9}$$

F1-Score

Utilizar apenas precisão transparece o desempenho do modelo em relação aos falsos positivos. Utilizar apenas revocação transparece o desempenho do modelo em relação aos falsos negativos. Para unir as vantagens dos dois modelos, a métrica *F1-Score* foi proposta na literatura, sendo sua equação descrita por:

$$F1 - Score = 2 \times \frac{Precisão \times Revocação}{Precisão + Revocação} \tag{2.10}$$

Capítulo 3

Trabalhos Relacionados

Este capítulo aborda os principais pontos da revisão de literatura feita para dar suporte as duas tarefas deste trabalho: classificação de títulos de notícias nas classes Petrobras, Vale, Itaú ou Outros; e classificação da movimentação do preço da ação PETR4. Para isso, o capítulo descreve artigos relacionados a classificação de textos, como também artigos que realizam predições dos preços de ações do mercado financeiro.

3.1 Classificação de textos

Artigos na área de categorização de notícias do mercado financeiro, classificação de críticas de filmes e detecção de sarcasmo foram revisados com o objetivo de auxiliar na Tarefa 1 deste trabalho. Schmitz *et al.* [5] contribuem e servem de inspiração para este trabalho ao criarem uma forma automática de coleta de notícias do mercado financeiro rotuladas nas seguintes categorias: negócios, mercado, internacional e economia bem como o treinamento de um modelo para a classificação dessas categorias (solução chamada de GOOSE). A motivação foi auxiliar o mercado financeiro com uma forma eficiente de gerar informações e *insights* relevantes para os investidores tomarem uma decisão sobre seus investimentos, seja de renda fixa ou variável. Isso ocorre pois, ao treinar um modelo que aprende a categorizar notícias nas classes mencionadas, facilita o trabalho do investidor em buscar por notícias de seu interesse. As notícias foram coletadas de diversos portais brasileiros de notícias do mercado financeiro, são eles: InfoMoney[1], MoneyTimes e Suno Research. Foram extraídas 143353 notícias desses portais com o objetivo de treinar um modelo de incorporação de palavras do tipo *GloVe*. Já para treinamento, foi utilizado um modelo de arquitetura *Bi-LSTM* e alimentado apenas com os títulos das notícias do portal Suno Research (19338 amostras). Desse valor, 80% foi utilizado para treinamento e 20% para teste. Os resultados mostram que o GOOSE obteve sucesso em classificar

[1] https://www.infomoney.com.br/

textos curtos - títulos de notícias - bem como gerar uma base de dados rotulada de forma automatizada. A acurácia média para predizer as quatro categorias foi de 84% e os dois maiores *F1-Scores* foi de 91% e 82% para as classes Negócios e Mercado, respectivamente.

A classificação de sentimentos em críticas de filmes é explorada por Alaparthi e Mishra [13] ao investigar a eficiência de quatro técnicas de análise de sentimento diferentes na classificação de 50 mil críticas de filmes, sendo as abordagens: não supervisionada (baseado no léxico das palavras, utilizando um dicionário), supervisionada com modelo tradicional (LR), supervisionada com modelo de aprendizado profundo (*LSTM*) e aprendizado profundo avançado (*BERT*). A base de dados foi separada em 35% para treinamento, 15% para validação e 50% para teste para o *BERT* e 60% para treinamento e 40% para teste nas demais abordagens. Os experimentos constataram uma superioridade do modelo *BERT* e regressão logística, com seus *F1-Scores* sendo de 93.2% e 89.4%, respectivamente.

Nayak e Bolla [14] também propõem uma análise comparativa de modelos na área de classificação, mas se diferencia de Alaparthi e Mishra [13] ao comparar modelos de redes neurais e representações realizadas por técnicas de extração de características de textos para encontrar a combinação mais eficiente na tarefa de detecção de sarcasmo em textos curtos. A metodologia consistiu nas seguintes etapas: (1) pre-processamento textual, (2) geração da incorporação de palavras, (3) classificação e (4) avaliação. Em cada etapa, diferentes combinações de modelos e representações foram avaliados. A fonte de dados do trabalho foi utilizada anteriormente por um outro artigo na literatura e extraída da plataforma *Kaggle*, sendo 13634 textos contendo sarcasmo e 14985 não contendo. Os autores descobriram que as melhores técnicas para extrair as características para a tarefa em questão são *BERT* e *word2vec*. Além disso, constataram também que *LSTM* e *Bi-LSTM* são os modelos mais promissores para classificar esses textos.

Outra análise comparativa foi feita por Sravya e Kumar [3], mas desta vez comparando 19 modelos diferentes de predição com o objetivo de classificar notícias em oito categorias, sendo as notícias estando na linguagem Telegu, comumente falado na Índia. Os modelos foram criados a partir da alternação entre os modelos de aprendizado de máquina *CNN*, *Bi-GRU*, *GRU*, *Bi-LSTM*, *LSTM* ou *SVM*; e as técnicas *word2vec*, *TF-IDF*, Contagem, *fastText* ou *GloVe* para a geração das representações numéricas dos dados textuais. A base de dados foi coletada no portal *Kaggle*. Os resultados dos experimentos mostram que as seguintes combinações foram as melhores: (1) o modelo *GRU* em conjunto com *fastText* ou *GloVe*, (2) *Bi-LSTM* com *word2vec* e (3) *SVM* com o *BoW* do tipo *TF-IDF* ou Contagem.

Similarmente a Sousa *et al.* [9], Zhang e Fucheng [4] também utilizaram o modelo de *CNN* adaptado para lidar com textos, o *textCNN*, mas os chineses exploraram a tarefa de categorizar dados textuais desbalanceados. Os dados textuais foram títulos de notícias

escritos em chinês, que compõem 10 categorias. Os dados foram extraídos do portal THUCNews. Para lidar com o desbalanceamento, o artigo propõe um método de *data augmentation* em que a sentença é convertida para inglês e convertida novamente para chinês, alterando a ordem inicial das palavras e criando dados novos para treinamento. O maior *F1-Score* para a predição de uma das categorias foi de 92.9% e o menor, de 68.7%.

3.2 Abordagens para a avaliação e precificação de uma ação

Foi percebido grande quantidade de artigos buscando realizar a tarefa de predição do preço de uma ação: por meio de regressão (para prever qual valor uma determinada ação vai ter em um determinado dia) ou classificação (para estabelecer se a ação vai ter seu preço diminuído, mantido constante ou aumentado). Para realizar essa tarefa, esses artigos utilizam dos conceitos de análise técnica, fundamental, ou a combinação de ambas, chamada neste trabalho de análise técnico-fundamental, para realizar seus experimentos.

3.2.1 Análise técnica

Mojtaba *et al.* [8] e Chung e Shin [7] utilizam unicamente da análise técnica para realizar as predições das ações. Mojtaba *et al.* [8] propõem um estudo comparativo entre nove modelos de aprendizagem de máquina e dois modelos de aprendizado profundo. Duas abordagens de valores de entrada para esses modelos foram adotadas: dados contínuos e dados binários. O estudo revelou que os dados binários possuem um desempenho melhor que dados contínuos e que os modelos de aprendizado profundo, *RNN* e *LSTM*, foram os melhores.

Chung e Shin [7] apresentam uma abordagem híbrida para prever o preço de uma ação por meio de regressão, integrando *LSTM* e um algoritmo genético. A metodologia foi avaliada em dados do *Korea Stock Price Index (KOSPI)*. Os resultados mostraram que o resultado híbrido atingiu um valor de 181.99 para a métrica *Mean Squared Error (MSE)*.

3.2.2 Análise fundamental

Considerando a utilização da análise fundamental, Sousa *et al.* [9] realizam uma análise comparativa entre os modelos *BERT*, *SVM*, NB e *textCNN*[2] para a classificação de sentimentos de notícias. Com isso, informações úteis são geradas com o objetivo de predizer

[2]CNN modificada por Yoon Kim para a tarefa de classificação de sentenças

valores do índice americano *Dow Jones Index (DJI)*. Para isso, 592 notícias de diversas fontes do ramo financeiro e relacionadas com o índice foram extraídas utilizando a ferramenta **Selenium**[3] e posteriormente foram rotuladas manualmente por quatro voluntários do grupo de pesquisa dos autores. Cada notícia foi rotulada como tendo o sentimento negativo, neutro ou positivo. Para comparar os modelos citados anteriormente, é feita uma alternância de *BoWs* do tipo Contagem e *TF-IDF* para os modelos tradicionais *SVM* e NB. Para avaliação desses modelos preditivos, foi utilizada a técnica *10-fold Cross-Validation* e o uso das métricas acurácia, precisão, revocação e *F1-Score*. Após as comparações, foi constatado que *Bidirectional Encoder Representations for Transformers (BERT)* foi o melhor em todas as métricas de avaliação, o modelo *textCNN* o segundo melhor em acurácia e precisão e *SVM* com *BoW* do tipo Contagem foi segundo melhor na revocação e *F1-Score*. Além disso, foi constatado que não houve correlação entre a taxa de notícias positivas ao longo do tempo e o valor do índice (*DJI*) ao longo desse mesmo tempo, sendo mais promissor utilizar os resultados dos sentimentos para classificar se o índice vai subir (sentimento positivo), ficar constante (sentimento neutro) ou descer (sentimento negativo). Ademais, foi percebido também que é difícil ter resultados satisfatórios prevendo o valor de um índice como o *DJI*, pois ele representa centenas de ações americanas. Como trabalho futuro, propuseram coletar notícias de ações em vez de índices e prever o valor de cada uma individualmente, proposta que o presente trabalho explora na Tarefa 2, ao prever o preço da ação PETR4.

Kalra e Prasad [10] expõem experimentos com objetivos similares a Sousa *et al.* [9], pois também buscam comprovar a hipótese de que o sentimento de notícias possui correlação com a movimentação do mercado financeiro. Com esse objetivo, avaliam a predição do comportamento de quatro ações de bancos indianos. Para isso, coletam dados históricos (preço de abertura, fechamento, maior preço, menor preço e volume) dessas ações, por meio do *Yahoo Finance*, e notícias relacionadas a eles. O pré-processamento dos dados textuais consistiu de tokenização, transformação em letra minúscula, remoção de *stopwords* e conversão para *BoW* do tipo *TF-IDF*. O trabalho experimenta os modelos *KNN*, *NB*, *SVM* e *ANN*. Para alimentá-los, foi utilizado apenas o atributo título de cada notícia; e para avaliá-los, utilizou-se a abordagem *Holdout* com 70% para treinamento e 30% para teste e as quatro métricas também abordadas nos trabalhos de Sousa *et al.* [9]. Os resultados mostram que *KNN* obteve o resultado mais satisfatório em dois dos quatro bancos, com 85% e 91% de acurácia, respectivamente. Para os outros dois bancos, os modelos *NB* (80% de acurácia) e *SVM* (81% de acurácia) foram os melhores.

Carosia et al. [11] também utilizam do sentimento em dados textuais para prever o comportamento de ações. Porém, diferente de Sousa *et al.* [9] e Kalra e Prasad [10],

[3]Biblioteca em linguagem Python para *crawling*, disponível em https://pypi.org/project/selenium

aprofundam no mercado financeiro brasileiro ao analisar o principal índice deste país, Ibovespa, utilizando de dados textuais em português. Além disso, utilizam de mais de uma fonte textual: *tweets* e notícias. Para realizar a predição, é proposto um modelo *ANN* do tipo *MLP* com arquitetura formada por 3 camadas e 10 neurônios. Para avaliá-la, 11027 *tweets* e 2132 notícias foram usadas para treinamento e 22236 *tweets* e 509 notícias para teste. Os resultados mostram que notícias têm mais influência sobre a variação do Ibovespa no preço de abertura do dia seguinte, enquanto os *tweets* têm mais influência no preço de fechamento do mesmo dia em que o *tweet* foi postado.

3.2.3 Análise técnica-fundamental

Com o objetivo de prever o valor de uma ação utilizando tanto a análise técnica quanto a fundamental, Ko e Chang [12] propõem o uso de duas arquiteturas de *deep learning*, *BERT* e *LSTM* para realizar a regressão do preço de ações do mercado tailandês. A base de dados para o experimento foi coletada levando em consideração quatro ações da bolsa de valores *Taiwan Stock Exchange* e foi dividida em três categorias: (1) dados históricos de cada ação em que contém os atributos: preço de abertura, fechamento, maior preço, menor preço e volume; (2) dados textuais de notícias nas classes finanças, política e notícias internacionais; e (3) dados textuais do fórum mais popular de Taiwan, chamado PTT. Para coletar as três categorias de dados, foi realizado uma raspagem de páginas web utilizando as bibliotecas **Requests**[4] e **Beautiful Soup**[5]. Os dados históricos e textuais foram utilizados para análise técnica e fundamental, respectivamente. A arquitetura *BERT* recebe os dados textuais para identificar os sentimentos dos textos, realizando uma análise fundamental; pois foi considerada a hipótese de que um sentimento negativo, neutro e positivo de um texto indica uma descida, inércia ou subida do seu valor, respectivamente. Já a segunda arquitetura recebe os dados rotulados com os sentimentos (saída do *BERT*), além de outras entradas relacionadas a dados históricos quantitativos para realizar uma regressão e prever os preços de abertura da ação no dia seguinte, praticando a análise técnica. Os experimentos mostram que utilizar dados textuais junto com dados históricos geram resultados melhores do que utilizar apenas dados históricos, havendo um aumento de 12.05% no *Root Mean Squared Error* (RMSE) ao comparar o desempenho dessas duas abordagens.

Renato *et al.* [1] aplicam os conhecimentos da análise técnica-fundamental ao analisar os fenômenos, regras e funcionamento da bolsa de valores B3 e agir no mercado utilizando um modelo composto por arquitetura *LSTM*. O modelo proposto (chamado de AURORA) é autônomo do tipo *agent-based model*, uma vez que simula o comportamento humano

[4]https://pypi.org/project/requests/
[5]https://pypi.org/project/beautifulsoup4/

ao decidir se o momento é de comprar, vender ou segurar as seguintes ações ordinárias: Petrobras (PETR3), Vale S.A. (VALE3) e Ambev (ABEV3). Para construir o solução, foi necessário as informações: nome da ação, código, tipo de mercado e informação histórica para compor a análise técnica. Esses dados foram obtidos do site da B3. Para realizar a análise fundamental, informações de múltiplas fontes foram coletadas para refletir a situação atual da ação no mercado. Para isso ser feito, o uso da ferramenta de busca de volumes *Google Trends*[6] foi utilizada. Os resultados do AURORA foram promissores, com acurácia de 82.96% no pior caso para o modelo prever o ganho ou perda do valor de uma ação, e 89.23% no melhor caso.

3.3 Considerações finais

Um resumo dos principais trabalhos é apresentado na Tabela 3.1. Nesse resumo, trabalhos que realizam a predição do preço de ações do mercado financeiro utilizando análise técnica e fundamental são identificados pelas siglas AT e AF, respectivamente, para uma melhor visualização. Observando a tabela, percebe-se vários trabalhos relacionados à categorização de textos curtos e na predição do comportamento do valor de uma ação do mercado financeiro. Porém, para o melhor do conhecimento obtido nesta revisão de literatura, não há trabalhos buscando categorizar notícias do mercado financeiro nas empresas que mais se referem, no caso deste trabalho, nas empresas Petrobras, Vale, Itau ou Outros. Além disso, não há trabalhos que buscam prever o comportamento da ação brasileira PETR4 por meio da análise fundamental, utilizando títulos de notícias. Portanto, tomando como principais inspirações as contribuições dos artigos Schmitz *et al.* [5], Kalra e Prasad [10] e Sravya e Kuma [3], este trabalho realiza experimentos relacionados à categorização de notícias e na predição do movimento do preço da ação PETR4, também se baseando em notícias. Para isso, utiliza alguns dos métodos que estão muito presentes na literatura para essas tarefas: *RNNs* do tipo *LSTM*, modelos de aprendizado de máquina tradicionais *KNN*, *SVM*, *NB* e *LR*, além das representações numéricas do tipo Contagem, *TF-IDF* e *word2vec*.

[6] https://trends.google.com.br/trends/

Tabela 3.1: Resumo dos estudos apresentados e as respectivas técnicas e resultados.

Autores	**Ano**	**Tarefa**	**Método**	**Resultados**
Mojtaba *et al.* [8]	2020	(AT) Classificação do preço de quatro grupos do mercado financeiro de Tehran	*LSTM* e *RNN*	*F1-Score*: 90%
Chung e Shin [7]	2018	(AT) Regressão do valor do índice KOSPI	*GA-LSTM*	*MSE*: 181.99
Sousa *et al.* [9]	2019	(AF) Classificação do comportamento do índice DJI	*BERT*	*F1-Score*: 72.5%
Kalra e Prasad [10]	2019	(AF) Classificação do comportamento de quatro ações indianas	*KNN*	*KNN*: 91%
Carosia *et al.* [11]	2019	(AF) Classificação do comportamento do índice Ibovespa	MLP	Acurácia: 60.6%
Ko e Chang [12]	2021	(AT e AF) Regressão do valor de ações de Taiwan	*BERT-LSTM*	*RMSE*: 1.8935
Renato *et al.* [1]	2022	(AT e AF) Prever o ganho ou perda do valor de ações utilizando um Modelo autônomo	*LSTM*	Acurácia: 89.23%
Schmitz *et al.* [5]	2022	Categorização de textos curtos derivados de múltiplas fontes de informação	*Bi-LSTM*	Acurácia: 84%
Alaparthi e Mishra [13]	2020	Classificação de sentimentos de críticas de filmes	*BERT*	Acurácia: 93.2%
Nayak e Bolla [14]	2022	Detecção de sarcasmo em textos curtos	*LSTM* e *Bi-LSTM*	Acurácia: 89.5%
Sravya e Kumar [3]	2022	Categorização de textos comparando modelos e representações numéricas	*fastText-GRU*	Acurácia: 70.92%
Zhang e Fucheng [4]	2021	Categorização de títulos de notícias chinesas com dados desbalanceados	*textCNN*	*F1-Score*: 92.9%

Capítulo 4

Metodologia

Este capítulo descreve em maiores detalhes os procedimentos aplicados para a realização dos experimentos de duas tarefas propostas neste trabalho: (1) Tarefa 1, que faz a classificação de notícias nas classes Petrobras, Vale, Itaú ou Outros; e (2) Tarefa 2, que busca classificar a movimentação do preço da PETR4. Além disso, é explicada a motivação para a criação do método que é utilizado para os experimentos das tarefas mencionadas anteriormente.

Para este trabalho, é considerada como a base de dados todos os arquivos que foram coletados das fontes mencionadas na Seção 4.1 e estão em seu estado bruto, sem processamentos adicionais. Já os dados produzidos a partir dos arquivos dessa base, que sofreram diversas manipulações e transformações para ser possível os experimentos deste trabalho, são chamados de *datasets*. Para realizar a Tarefa 1, é montado um *dataset* rotulado a partir de dados qualitativos. No caso da Tarefa 2, também é montado um único *dataset* rotulado, mas a partir de dados qualitativos e quantitativos.

4.1 Base de dados

A base de dados utilizada para a realização deste trabalho é composta por dados qualitativos e quantitativos. Os dados qualitativos são dados textuais de notícias coletados por Schmitz *et al.* [5] dos portais Suno Research e Money Times a partir de *crawlers*[1] [5], e disponibilizados em um repositório público[2]. Os dados contemplam notícias postadas entre os dias 28/07/2020 e 03/08/2022, período da postagem das notícias coletadas pelos *crawlers*, e estão armazenados em arquivos no formato **JSON**. Entre os arquivos do repositório mencionado anteriormente, estão arquivos contendo notícias relacionadas as empresas Petrobras, Itaú e Vale e um arquivo contendo notícias relacionadas a entidades

[1] Algoritmo que possui o objetivo de extrair informações de páginas *web* de forma automatizada
[2] https://github.com/mso13/Mestrado_PPGI/tree/main/src/crawlers

diversas. Esses arquivos contêm informações sobre o tópico (uma única palavra chave que resume o contexto da notícia), título da notícia, data de postagem da notícia, data que o *crawler* extraiu a notícia, *link* da notícia e suas *tags* (palavras-chave que resumem do que a notícia se trata), cujos nomes dos seus respectivos atributos no arquivo **JSON** são: "*topic*", "title", "*date*", "*search_date*", "*url*" e "*tags*". Para fins de organização da base de dados deste trabalho, esses arquivos foram armazenados em um repositório[3] público, de autoria própria.

Em relação aos dados quantitativos, são dados históricos da da ação PETR4 extraídos do portal Yahoo Finance[4]. A extração foi possível pois o portal possibilita selecionar um período para os dados históricos e disponibiliza um *link* para seu *download*. Os dados contêm as movimentações da ação entre os dias 21/09/2020 e 02/08/2022. O período considerado foi semelhante ao período utilizado nos dados qualitativos, com o objetivo de facilitar a criação do *dataset* da Tarefa 2. O motivo dessa decisão é explicado em mais detalhes na Subseção 4.4.1. Essas informações foram armazenadas em um único arquivo, em formato **CSV**. Esse arquivo também está armazenado no repositório público de autoria própria (mesmo repositório em que os dados qualitativos foram armazenados). O arquivo contém informações sobre o preço de abertura da ação, preço máximo, preço mínimo, preço de fechamento, preço ajustado (preço que inclui influência da distribuição de proventos da ação), volume (quantidade de transações de compra ou venda da ação) e a data que esses preços foram estabelecidos para cada dia. Os nomes das colunas dessas informações no arquivo **CSV** são, respectivamente, "*Open*", "*High*", "*Low*", "*Close*", "*Adj Close*" e "*Volume*" e "*Date*".

4.2 Classificar notícias nas classes Petrobras, Vale, Itaú ou Outros (Tarefa 1)

A Tarefa 1 é responsável pela categorização supervisionada de títulos de notícias do mercado financeiro nas seguintes classes: Petrobras (Classe 1), Vale (Classe 2), Itaú (Classe 3) ou Outros (Classe 0). Por esta tarefa não se tratar de predições do preço de ações, as análises descritas na Seção 2.1 não são utilizadas. Em relação à base de dados, apenas os dados qualitativos são utilizados. Isso se deu pelo fato de apenas notícias serem utilizadas para a criação do *dataset*. Os detalhes das características desses dados qualitativos estão explicados na Seção 4.1. Foi percebido no trabalho de Renato *et al.* [1] a escolha de três empresas do mercado financeiro de setores diferentes ao fazer predições dos preços de suas ações. Apesar do trabalho não informar se essa decisão foi proposital ou não, este traba-

[3]https://github.com/guico3lho/DataScience_Assets
[4]https://finance.yahoo.com/quote/PETR4.SA/history?p=PETR4.SA

lho também escolhe empresas de setores diferentes da economia, sendo as empresas das Classes 1, 2 e 3, relacionadas aos setores de Petróleo, Minério e Banco, respectivamente, presumindo que essa decisão contribuiria para que o aprendizado dos modelos preditivos seja mais satisfatório. Caso existisse duas classes contendo dois bancos, Itaú e Bradesco por exemplo, era esperado que o modelo preditivo teria dificuldade em diferenciar uma classe da outra.

Para fazer a rotulagem das classes, a coluna "tags" foi utilizada. Caso a palavra-chave "Petrobras" esteja entre as palavras-chave dessa coluna, a notícia é rotulada como Petrobras (Classe 1), e assim por diante nas Classes 2 e 3. Já para notícias que não contenham nenhuma das palavras-chave que identificam as classes das empresas (Petrobras, Vale e Itaú), são rotuladas como Outros (Classe 0). Portanto, a presença da Classe 0 é importante para que o modelo aprenda a identificar notícias que não sejam relacionadas a nenhuma das três empresas citadas.

4.3 Classificar a movimentação do preço da PETR4 (Tarefa 2)

A Tarefa 2 utiliza dos conceitos de análise fundamental para prever o movimento do preço da ação PETR4, da empresa Petrobras. A decisão por prever o preço da ação dessa empresa se deu por conta da ser uma empresa muito presente para a economia brasileira. Para esta tarefa, que também é supervisionada, é predito se o movimento do preço da ação Desceu (Classe 0), permaneceu Constante (Classe 1) ou Subiu (Classe 2). As regras consideradas para definir essas classes são descritas na Subseção 4.4.1. Para essa tarefa, os dados qualitativos e quantitativos da base de dados são utilizados. Esses dois tipos de dados são utilizados pois a criação do *dataset* desta tarefa utiliza tanto notícias (dados qualitativos) quanto dados históricos da ação PETR4 (dados quantitativos). No caso dos dados qualitativos, foram apenas considerados os arquivos **JSON** contendo notícias da Petrobras. Para os dados quantitativos, foi utilizado o único arquivo **CSV** que está presente na base de dados, contendo os dados históricos das movimentações do preço da ação PETR4. Os detalhes das características desses dados quantitativos e qualitativos estão explicados na Seção 4.1.

4.4 Método de classificação de títulos de notícias

Ao longo da implementação das duas tarefas, apresentadas nas Seções 4.2 e 4.3, realizadas neste trabalho, foi percebido que ambas compartilham as seguintes etapas: (1) Preparação

do *dataset*, (2) Normalização dos textos, (3) Extração de características, (4) Otimização dos hiperparâmetros dos modelos preditivos e (5) Avaliação dos modelos otimizados. Portanto, foi definido um método de classificação de títulos de notícias a partir dessas etapas, cuja a estrutura pode ser visualizada no *pipeline* da Figura 4.1. Ademais, por ter sido observada na revisão de literatura (Capítulo 3) deste trabalho a utilização das técnicas de extração de características do tipo Contagem e *TF-IDF* apenas em modelos tradicionais e a técnica *word2vec* apenas em modelos de *deep learning*, este trabalho realiza os experimentos também considerando esse raciocínio. Além dessa diferenciação da extração de características dos dois tipos de aprendizados, a sua otimização também precisou ser diferente. Portanto, devido às diferenciações percebidas, as Etapas (2) e (3) do método proposto são aplicadas de forma separada para os modelos tradicional e de *deep learning*, sendo explicadas em mais detalhes nas Subseções 4.4.3 e 4.4.4, respectivamente. As seções seguintes descrevem em detalhes, considerando as duas tarefas realizadas neste trabalho, as etapas do método proposto.

4.4.1 Preparação dos *datasets*

Esta etapa do método é responsável por receber como entrada os dados brutos, apresentados na Seção 4.1, considerados para cada tarefa. A saída desta etapa são dois *datasets* (contendo apenas as colunas "title" e "label"), um para cada Tarefa, que serão utilizados ao longo de todo o *pipeline* (Figura 4.1). Os dois *datasets* foram separados, de forma aleatória e estratificada, em *splits* de treinamento, validação e teste, com as porcentagens de 80%, 10% e 10% para cada divisão, respectivamente. Cabe destacar que a mesma porcentagem desses *splits* será compartilhada tanto pelo fluxo de aprendizado tradicional quanto de *deep learning*, com o objetivo de tornar justa a comparação entre os resultados dos dois aprendizados. Além disso, no início da preparação dos dois *datasets*, foi feita uma conversão dos arquivos **JSON** e **CSV** para o formato de tabela, utilizando a biblioteca **Pandas**[5], com objetivo de facilitar futuras manipulações e armazenamento dos dados processados. Portanto, o termo "tabela" passará a ser utilizado, em vez do termo "arquivo".

Preparação do *dataset* da Tarefa 1

Conforme discutido na Seção 4.2, apenas os dados qualitativos (dados textuais, no caso deste trabalho) da base de dados (4.1), foi considerada para a Tarefa 1. Foi necessária as tabelas contendo notícias das empresas Petrobras, Vale e Itaú e também a tabela contendo notícias de assuntos diversos. As tabelas contendo as notícias das empresas foram filtradas

[5]https://pandas.pydata.org/

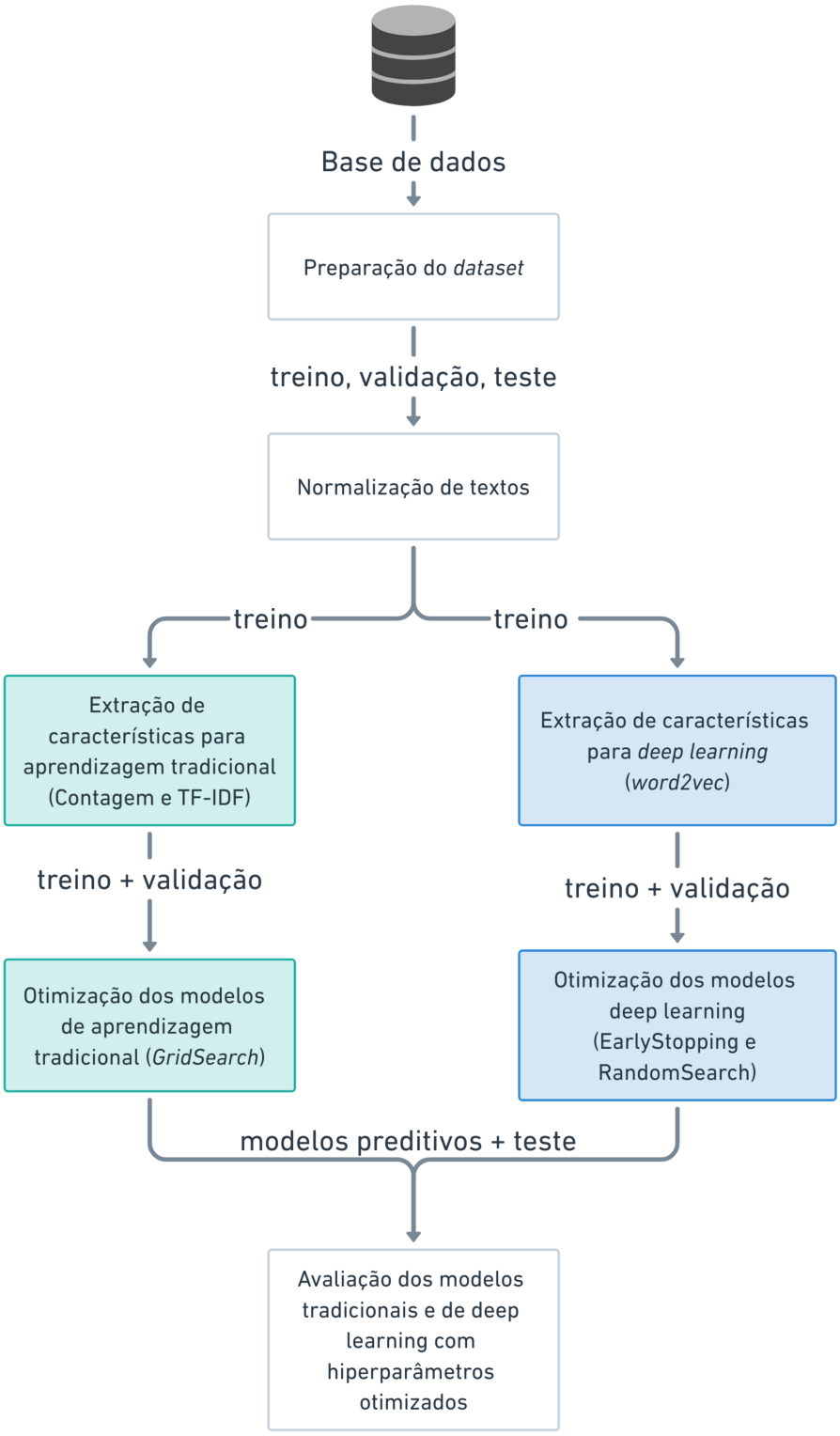


Figura 4.1: Diagrama contendo o *pipeline* para o método proposto para a realização dos experimentos das Tarefas 1 e 2 (fonte: autoria própria).

para que contenham apenas notícias relacionadas a apenas um dos rótulos. A tabela contendo notícias de entidades diversas também foi filtrada para que contenha apenas notícias não relacionadas a nenhuma das classes que representam as empresas citadas. Isso foi feito para ser possível a criação do *split* representando a Classe 0 (Outros). As filtragens puderam ser realizadas por conta da coluna "tags". Posteriormente, as tabelas foram rotuladas com suas respectivas classes e concatenadas para formar o *dataset*, composto pelas colunas "title" e "label". A coluna "title" é o único atributo do *dataset* e a coluna "label" é usada para que o aprendizado supervisionado seja possível.

O *dataset* criado possui um total 18012 amostras e duas colunas ("title" e "label"). A quantidade de amostras para cada classe e para cada *split* pode ser visto na Tabela 4.1. As colunas que estão contidas nesse *dataset* estão apresentadas em uma amostra do *dataset*, representado pela Tabela 4.2.

Tabela 4.1: Quantidade de amostras para cada classe e porcentagem de cada *split*.

Conjunto	**Divisão (%)**	**Petrobras (1)**	**Vale (2)**	**Itaú (3)**	**Outras (0)**
Treino	80	4887	2994	1714	4814
Validação	10	588	367	225	621
Teste	10	638	379	220	565
Total	100	6113	3740	2159	6000

Tabela 4.2: Amostra do *Dataset* criado para a Tarefa 1.

	title	*label*
0	A Gol (GOLL4) corta 35% do salário de seus funcionários	0
1	Dividendos da BR Distribuidora (BRDT3), Inter (BIDI11) e Petrobras (PETR4) são destaques	1
2	Embarque de minério do Brasil sobe 24% em volume em maio; mais que dobra em valor	2
3	Ibovespa tem leve alta, mas é contido por commodities; SulAmérica (SULA11) dispara 7%	2
4	Itaú é o mais bem preparado para as mudanças digitais, avalia Santander	3
5	Ivan Monteiro pode ir para Banco do Brasil, diz Bolsonaro	1
6	M. Dias Branco recomprará até 10% das ações em circulação	0
7	Marfrig, JBS e Minerva são autorizadas a exportar para Tailândia	0
8	Petrobras (PETR4) diz que alta do petróleo indica necessidade de reajuste de combustíveis	1
9	Sede da Petrobras na Bahia custou 4 vezes mais, diz MPF	1

Preparação do *dataset* da Tarefa 2

Já para a criação do *dataset* da Tarefa 2, a tabela contendo os dados históricos da ação da PETR4 foram concatenados com a tabela contendo apenas notícias relacionadas a Petrobras. Essa concatenação foi possível pelo fato das duas tabelas possuírem a coluna "data" em comum. Dessa forma, foi feito um *join* utilizando essa coluna como chave. Para ser possível a criação da coluna "*label*", que contém os rótulos das classes desta tarefa, primeiramente foi necessário criar uma coluna chamada "*profit*" em que foram utilizadas as colunas dos dados históricos "*Open*" (preço de abertura da ação no dia) e "*Close*" (preço de fechamento da ação no dia) e satisfazendo a Equação 4.1.

$$profit = \frac{Close - Open}{Open} \tag{4.1}$$

Assim, essa coluna foi utilizada para a criação da coluna "label", regida pela seguinte regra:

- Se "*profit*" < -0.25%, "*label*" = 0 (desceu);
- Se |"*profit*"| <= 0.25%, "*label*" = 1 (constante);
- Se "*profit*" > 0.25%, "*label*" = 2 (subiu).

Cabe destacar que, para esta tarefa, o método para a criação das colunas *profit* e *label*, bem como a porcentagem (chamada de *threshold*) utilizada para decidir a classe a ser predita são propostos por Medeiros [27]. Na Figura 4.2 é ilustrado um exemplo de amostra em que o atributo "profit" foi criado e na Figura 4.3 ilustra essa mesma amostra, mas com o rótulo definido como 0, pois |"*profit*"| < 0.25%. Além disso, para tornar mais fidedigno a relação que as notícias têm com o valor de fechamento do dia, foram consideradas apenas notícias que foram postadas no horário até as 18:15, período em que a bolsa brasileira fecha para transações. Portanto, notícias postadas depois desse horário terão influência apenas no preço de abertura e fechamento do dia seguinte. Ao final de todas as manipulações desta etapa, o *dataset* é constituído pelas colunas "title", coluna empregada para realizar as predições e "label", contendo os rótulos das classes desta tarefa.

	Date	Open	High	Low	Close	Adj Close	Volume	profit_brute	profit
0	2020-07-16	23.160000	23.280001	22.520000	22.719999	10.791756	69698700	-0.440001	-0.018998

Figura 4.2: Amostra contendo o resultado do cálculo da coluna *profit* (fonte: autoria própria).

date	title	tags	url	time	label
2020-07-16	OGCI estabelece nova meta de emissão de carbon...	[Combustíveis, Empresas, Meio Ambiente, Petrob...	https://www.moneytimes.com.br/ogci-estabelece-...	10:15:00	0

Figura 4.3: Amostra contendo a formação do rótulo (fonte: autoria própria).

O *dataset* criado possui um total de 4431 amostras e duas colunas ("title" e "label"). A quantidade de amostras para cada categoria e para cada *split* pode ser visto na Tabela 4.3. Além disso, as colunas que estão contidas nesse *dataset* estão apresentadas em uma amostra representada na Tabela 4.4.

Tabela 4.3: Quantidade de amostras de cada tipo de movimentação da ação e porcentagem de cada *split*.

Conjunto	**Divisão (%)**	**Desceu (0)**	**Constante (1)**	**Subiu (2)**
Treino	80	1678	354	1512
Validação	10	221	46	176
Teste	10	214	35	195
Total	100	2113	435	1883

Tabela 4.4: Amostra do *Dataset* criado para a Tarefa 2.

	title	*label*
0	Carteira recomendada da Mirae ganha 3 novas ações em abril; veja as novidades	0
1	Petrobras (PETR4): ações seguem atrativas, apesar da queda de hoje, diz banco; entenda por quê	0
2	Raízen também suspende vendas de gasolina de aviação após alerta da Petrobras	0
3	De JBS a Magazine Luiza: BB Investimentos indica 8 novas ações em carteira para janeiro	2
4	As 6 empresas que terão os melhores resultados do 1º trimestre, segundo o Santander	0
5	Chinesa CNODC optou por não comprar fatia adicional em Búzios, diz Petrobras	2
6	Petrobras (PETR4) sobe com fala de Bolsonaro sobre preço de combustível	0
7	Ibovespa: De olho em mudança na Petrobras (PETR3), ação é uma das mais negociadas	2
8	Leilão de petróleo tem recorde de R$ 8,9 bilhões e grupo Total bate Petrobras	2
9	Ibovespa tenta rescaldo, mas tensão com EUA e China limita recuperação	2

4.4.2 Normalização dos textos

Esta etapa recebe como entrada os *splits* dos dados (Subseção 4.4.1) e entrega como saída os dados textuais normalizados, utilizando técnicas descritas na Subseção 2.3.1, com exceção das técnicas de Lematização e Stemização. A coluna "title", apresentada nas Tabelas 4.2 e 4.4, contém os dados textuais que serão normalizados. A Figura 4.4 ilustra as etapas de normalização de texto aplicadas bem como a ordem seguida.

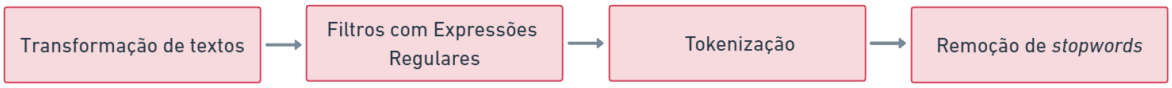


Figura 4.4: Normalização de textos utilizada pelo método proposto (fonte: autoria própria).

Em ambas tarefas foi realizado uma normalização textual similar. Essa normalização seguiu o fluxo ilustrado na Figura 4.4. As transformações feitas foram: conversão do texto em *lowercase*; conversão de símbolos em palavras. Por exemplo. "+" vira "mais" e "%" vira "por cento"; conversão de *emojis*, remoção de grandes espaços vazios entre as palavras e também conversão de números em "<NUM>", remoção de pontuações (utilizando uma biblioteca interna do Python, chamada **String**) e acentos (utilizando a biblioteca **Unidecodedata**[6]). Em relação ao filtro de expressões regulares, o padrão "`\b[a-zA-Z{3}[a-zA-Z0-9]{3,}\b`" foi utilizado. Esse padrão é um filtro que deixa passar apenas palavras cujo tamanho, em letras (maiúsculas ou minúsculas), seja de pelo menos 3 caracteres. A decisão por esse filtro se deu pela hipótese de que, no geral, palavras relevantes (que agregam mais sentido ao texto), contém pelo menos três letras. Dessa forma, esse filtro contribui com a remoção de *stopwords* ao também remover palavras que são consideradas ruídos. A tokenização foi feita considerando *ngrams* de uma palavra. Em seguida, foram removidas *stopwords* (utilizando a biblioteca ***Natural Language Toolkit (NLTK)***[7]). Ao final desta etapa, os textos da coluna "title" estão normalizados para todos os *splits*. A extração de características e otimização dos hiperparâmetros dos modelos de aprendizado tradicional e de *deep learning* é realizada em seguida, nas Subseções 4.4.3 e 4.4.4, respectivamente.

4.4.3 Aprendizado tradicional

Esta seção aborda a extração de características aplicando as técnicas Contagem e *TF-IDF* (explicadas na Subseção 2.3.2) nos textos, para que possam ser utilizados para o treina-

[6]https://docs.python.org/3/library/unicodedata.html
[7]https://www.nltk.org/

mento dos modelos tradicionais *KNN*, *SVM*, *NB* e *LR*. Além disso, expõe os procedimentos feitos para a otimização dos hiperparâmetros desses modelos.

Extração de características (Contagem e *TF-IDF*)

Esta etapa recebe como entrada apenas o *split* de treinamento; e como saída, as representações numéricas do tipo Contagem (Equação 2.1) e *TF-IDF* (Equação 2.2), conforme ilustrado na Figura 4.5. É importante que, nesta fase, apenas o *split* de treinamento seja utilizado para a extração de características e criação do vocabulário, para que o modelo preditivo não fique com viés e "roube" ao classificar os *splits* de validação e teste. Após serem treinados em cima do *split* citado, essas representações criam um vocabulário do *corpus* e a capacidade de extrair as características de textos ao transformá-los em vetores numéricos.

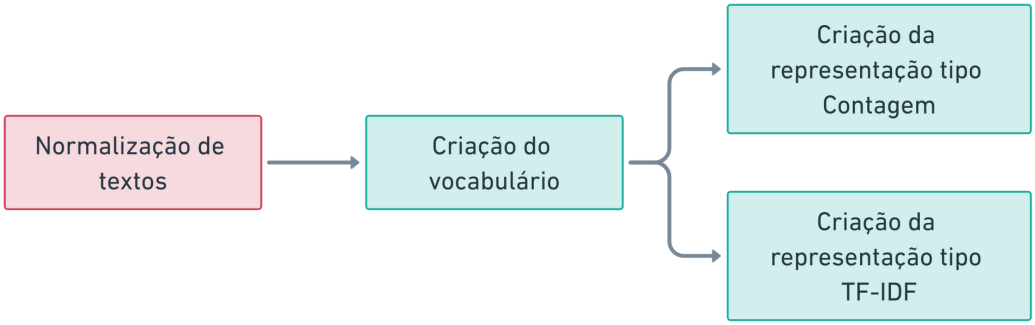


Figura 4.5: Fluxo da extração de características para aprendizado tradicional (fonte: autoria própria).

As representações numéricas do tipo Contagem e *TF-IDF* foram implementadas utilizando a biblioteca **Sci-kit Learn**[8]. Ambas foram criadas considerando *ngrams* de tamanho um, ou seja, o vocabulário das representações será formado por *tokens* constituídos de apenas uma palavra. Além disso, foi considerado que as palavras que estão presentes em mais de 50% dos documentos são ignoradas para a formação do vocabulário. Essa decisão se deu para remover palavras muito frequentes em todo o *corpus*. A quantidade de palavras do vocabulário das representações Contagem e *TF-IDF* e exemplos de suas representações numéricas para as Tarefas 1 e 2 estão detalhadas nas Subseções 5.2.2 e 5.3.2, respectivamente.

[8]https://scikit-learn.org/

Tabela 4.5: Conjunto de hiperparâmetros para otimização dos modelos tradicionais.

Modelo	**Hiperparâmetro**	**Argumentos**
KNN	n_neighbors	15, 17, 19, 21, 23, 25, 27
	weights	distance, uniform
	metric	cosine, eucliedean, manhathan
SVM	C	1, 5, 10
	kernel	rbf, linear
NB	alpha	0.1, 1, 10
	fit_prior	True, False
LR	penalty	l1, l2, None
	C	0.1, 1, 10
	solver	liblinear, sag, saga

Tuning dos modelos (GridSearchCV)

Esta etapa recebe como entrada as representações Contagem e *TF-IDF*, compostas por vetores numéricos, já treinadas no *corpus* do *dataset*, bem como os *splits* de treinamento e validação (apresentados nas Tabelas 4.1 e 4.3). A saída são os modelos tradicionais (*KNN*, *SVM*, *NB* e *LR*) com os melhores hiperparâmetros para cada representação, resultado apresentado nas Subseções 5.2.2 e 5.3.2, totalizando oito modelos (quatro para a representação Contagem e quatro para a representação *TF-IDF*). Para realizar a otimização, foi aplicada a função **GridSearchCV** (para implementar a técnica *holdout*) da biblioteca **Sci-kit Learn**[9] nos modelos preditivos, considerando os hiperparâmetros e argumentos descritos pela Tabela 4.5. Além disso, a otimização foi feita a partir dos *splits* de treinamento e validação (apresentados nas Tabelas 4.1 e 4.3) com a utilização da métrica acurácia (Equação 2.7) para decidir os melhores hiperparâmetros para cada modelo de aprendizado tradicional.

4.4.4 Aprendizado profundo

Esta seção apresenta em detalhes a realização da extração de características aplicando a técnica *word2vec* (explicada na Subseção 2.3.2) nos textos para que possam ser utilizados para o treinamento dos modelos de *deep learning LSTM* e *Bi-LSTM*. Além disso, apresenta os procedimentos feitos para a otimização dos hiperparâmetros desses modelos.

Extração de características (*word2vec*)

Diferente da extração de características para aprendizado tradicional, será utilizada a técnica *word2vec*, que implementa a representação *WordEmbeddings*. Similarmente à aborda-

[9]https://scikit-learn.org/

gem com o aprendizado tradicional, também será utilizado apenas o *split* de treinamento como entrada, mas a saída vai gerar uma representação *word2vec*, conforme ilustrado na Figura 4.6, em vez das representações Contagem e *TF-IDF*. Essa representação armazena o vocabulário do *corpus* do *split* de treino, bem como um vetor numérico para cada palavra, cuja função é armazenar seu contexto. A quantidade de palavras do vocabulário da representação *word2vec* e exemplos de sua representação numérica para as Tarefas 1 e 2 estão detalhadas nas Subseções 5.2.3 e 5.3.3, respectivamente.

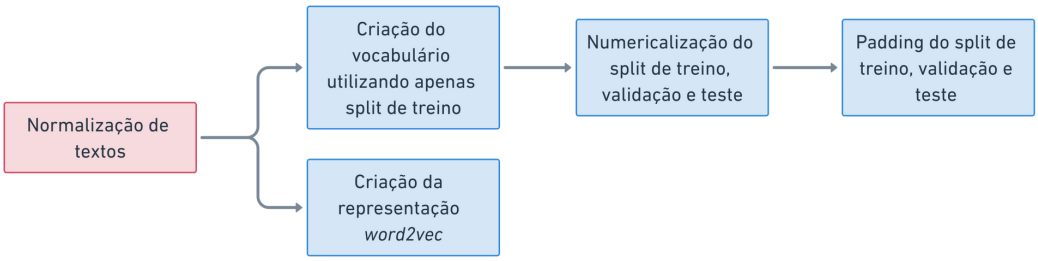


Figura 4.6: Fluxo de extração de características para aprendizado profundo (fonte: autoria própria).

Com a representação *word2vec* criada, é aplicada a classe **Tokenizer**, da biblioteca **Keras**[10], para treinar um tokenizador considerando o *corpus* do *split* de treinamento. Com o tokenizador treinado, ele também é aplicado nos demais *splits* (validação e teste), transformando todos os *splits* em formato de vetores numéricos. Posteriormente, foi aplicado um *padding* para que todos esse vetores possuam um único tamanho. Para encontrar esse tamanho, também foi considerado apenas o *split* de treino, e posteriormente aplicado nos demais *splits*. O cálculo desse tamanho foi feito ao aplicar a média dos tamanhos de todos os vetores numéricos do *corpus* de treinamento.

Após a criação do modelo *word2vec* e os dados textuais convertidos em vetores numéricos com *padding*, os dados estão preparados para servirem como entrada dos modelos de *deep learning*.

Tuning dos modelos (RandomSearch e Early Stopping)

Esta etapa recebe como entrada a representação *word2vec* e os dados textuais em formato de vetores numéricos. A saída são os modelos *deep learning LSTM* e *Bi-LSTM* com os hiperparâmetros otimizados, resultado apresentado nas Subseções 5.2.3 e 5.3.3. Para realizar essa otimização, as funções **Early Stopping** e **RandomSearch** foram consideradas. É importante ressaltar que a função **Random Search** possui a mesma função do **GridSearchCV**, que é a de implementar a técnica *holdout*, mas a troca foi motivada

[10]https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer

devido ao fato de não ter sido possível aplicar a função *GridSeachCV* para a otimização de modelos de *deep learning*. O uso do **Early Stopping** tem a função de interromper o treinamento da rede quando esta não diminui o valor da função de perda ou acurácia. No caso deste trabalho, foi determinado que a rede neural pararia seu treinamento quando a função de perda não diminuir em mais de dez épocas. Os argumentos considerados para a otimização de cada hiperparâmetro dos modelos *LSTM* e *Bi-LSTM* estão descritos na Tabela 4.6. Os *splits* de treinamento e validação foram utilizados para encontrar os melhores hiperparâmetros para os modelos de *deep learning*, levando a métrica acurácia (Equação 2.7) em consideração.

Tabela 4.6: Conjunto de argumentos para cada hiperparâmetro a ser otimizado pela *LSTM* e *Bi-LSTM*.

Hiperparâmetro	**Argumentos**
Neurônios na primeira camada (*LSTM* ou *Bi-LSTM*)	[16, 32, 64]
Taxa de esquecimento primeiro Dropout	[0, 0.1, 0.2, 0.3]
Neurônios na segunda camada (*LSTM* ou *Bi-LSTM*)	[16, 32, 64]
Neurônios na camada densa	[16, 32, 64]
Função de ativação da camada densa	[tanh, ReLU]
Taxa de esquecimento segundo Dropout	[0, 0.1, 0.2, 0.3]
Otimizador	[adam, RMSprop, SGD]

4.4.5 Avaliação dos modelos otimizados

Esta etapa recebe como entrada os modelos de aprendizagem tradicional e *deep learning*, ambos com hiperparâmetros otimizados (Subseções 4.4.3 e 4.4.4, respectivamente), e as amostras de teste, em representação numérica, das Tarefas 1 e 2 (Seções 4.1 e 4.3, respectivamente). Esta etapa é importante para os modelos terem contato com dados ainda não vistos, simulando um caso real em que seriam aplicados em uma pesquisa ou produto. A avaliação do desempenho dos modelos preditivos que estão nesta etapa se dá por meio das métricas acurácia, precisão, revocação e *F1-Score* (explicadas anteriormente na Subseção 2.5). Devido ao fato da métrica "acurácia" ter sido levada em consideração na etapa de otimização dos modelos, esta métrica também será considerada para decidir qual modelo teve o melhor desempenho. Além disso, por conta do fato dessa métrica contemplar apenas a taxa de acertos em relação ao total de predições e não transparecer o desempenho do modelo na classificação de falsos positivos e negativos, a métrica *F1-Score* também será considerada para a decisão. Porém, em caso de empate, ou seja, um modelo obteve a melhor acurácia e um outro obteve o melhor *F1-Score*, o desempate se dá pelo modelo que tiver a maior acurácia. Decidiu-se por essa métrica como desempate pelo fato de apenas

ela ter sido aplicada para a otimização desses modelos. Os resultados experimentais a partir dessas métricas são apresentados no Capítulo 5.

4.5 Considerações finais

Este capítulo expõe em detalhes as características da base de dados e do método criado para a realização das Tarefas 1 e 2 deste trabalho. Das etapas que compõe o método, apenas a explicação da Preparação do *dataset* foi diferente para cada a tarefa, sendo as demais etapas possuindo poucas mudanças entre as tarefas. Porém, no Capítulo 5, os resultados experimentais de todas as etapas executadas são distintos, devido ao fato de se tratar de duas tarefas com objetivos e *datasets* diferentes.

Capítulo 5

Resultados Experimentais

Este capítulo expõe o ambiente de desenvolvimento que foi utilizado para realizar os experimentos das etapas deste trabalho, apresentadas no *pipeline* da Figura 4.1, bem como seus resultados. São abordados os resultados e análises sobre os *datasets* criados, normalização dos dados textuais, extração das suas características e desempenho na otimização e avaliação dos modelos preditivos.

5.1 Ambiente de Desenvolvimento

Os experimentos foram realizados no sistema operacional Windows 10, sendo o ambiente de programação realizado na *Integrated Development Environment (IDE) PyCharm*[1]. Essa *IDE* possibilita a adição do *framework* Jupyter Notebook[2] em sua ferramenta, que também foi utilizado para os experimentos. Os códigos construídos para realizar todos os experimentos e geração de resultados das Tarefas 1 e 2 podem ser acessados em dois repositórios públicos de autoria própria[3][4]. As seguintes configurações de *hardware* foram empregadas para realizar todos os experimentos deste trabalho:

- Processador: Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz 3.20 GHz;
- Memória RAM: 12 GB;
- Placa de Vídeo: GeForce GTX 1060 Ti.

[1]https://www.jetbrains.com/lp/pycharm-pro
[2]https://jupyter.org
[3]https://github.com/guico3lho/TCC_tags
[4]https://github.com/guico3lho/TCC_stock_price

5.2 Resultados experimentais da Tarefa 1

Esta seção é responsável por mostrar os principais resultados obtidos da normalização textual, extração de características, otimização dos hiperparâmetros e avaliação dos modelos, etapas explicadas no Capítulo 4, a partir do *dataset* criado (Seção 4.2) para a Tarefa 1.

5.2.1 Normalização textual

Após a normalização dos textos da coluna "*title*" do *dataset* (Subseção 2.3.1), foi gerada uma nuvem de palavras para cada categoria com o objetivo de descobrir as palavras mais frequentes para cada classe do *dataset*. Quanto maior o tamanho da palavra na nuvem de palavras, mais frequente ela é no *corpus* que representa sua classe.

Na Figura 5.1, palavras como "petr4", "petroleo", "gas", "gasolina" e "diesel" são bastante frequentes; portanto, espera-se que o modelo preditivo classifique a notícia como sendo da classe Petrobras (Classe 1), caso receba títulos com essas palavras na entrada do modelo. A mesma ideia pode ser aplicada às demais classes. No caso da nuvem de palavras da Vale (Classe 2), ilustrada na Figura 5.2, as palavras minério, ferro, brumadinho, dividendo são mais aparentes. No caso do Itaú (Classe 3), as palavras "itub4", "xp", "carteira", "banco", "bradesco" e "brasil" aparecem bastante. Por fim, observa-se as palavras alta, queda, brasil e dólar nas notícias classificadas como Outros (Classe 0). Ao se observar nessas nuvens palavras que diferenciam uma classe da outra; é esperado que os modelos preditivos aprendam a classificar cada classe e possuam um desempenho satisfatório na etapa de avaliação.

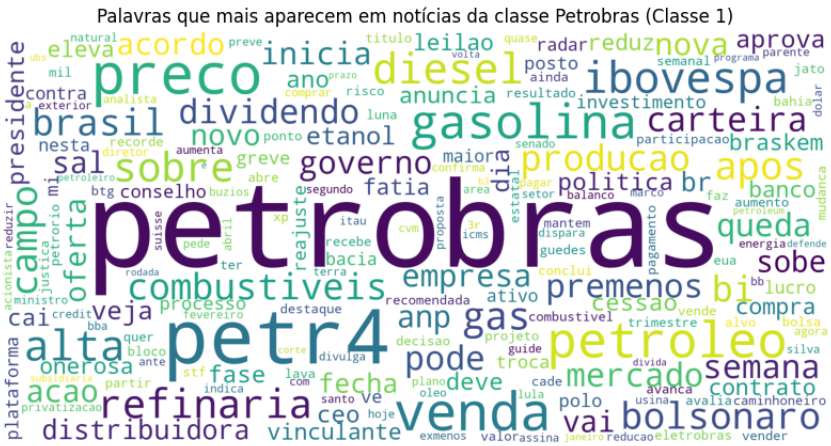


Figura 5.1: Nuvem com palavras mais recorrentes dos títulos de notícias classificados como Petrobras (Classe 1) (fonte: autoria própria).

Figura 5.2: Nuvem com palavras mais recorrentes dos títulos de notícias classificados como Vale (Classe 2) (fonte: autoria própria).

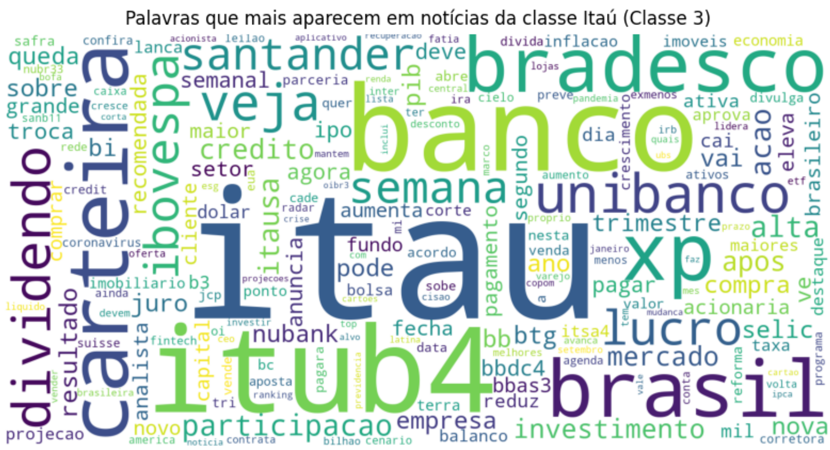


Figura 5.3: Nuvem com palavras mais recorrentes dos títulos de notícias classificados como Itaú (Classe 3) (fonte: autoria própria).

5.2.2 Aprendizado tradicional

Esta seção apresenta os principais resultados das etapas de extração de características e otimização dos modelos de aprendizado tradicional (Seção 4.4.3) da Tarefa 1.

Extração de características (Contagem e *TF-IDF*)

O treinamento das representações numéricas do tipo Contagem (Equação 2.1) e *TF-IDF* (Equação 2.2) no *split* de treino resultou em 4274 palavras constituindo seu vocabulário. Duas amostras dos textos normalizados são apresentadas (Frase 0 e Frase 1). Além disso, também é apresentado as representações Contagem e *TF-IDF*, ilustradas nas Figuras 5.5 e 5.6, respectivamente. A partir da observação dessas duas figuras e do que foi explicado na

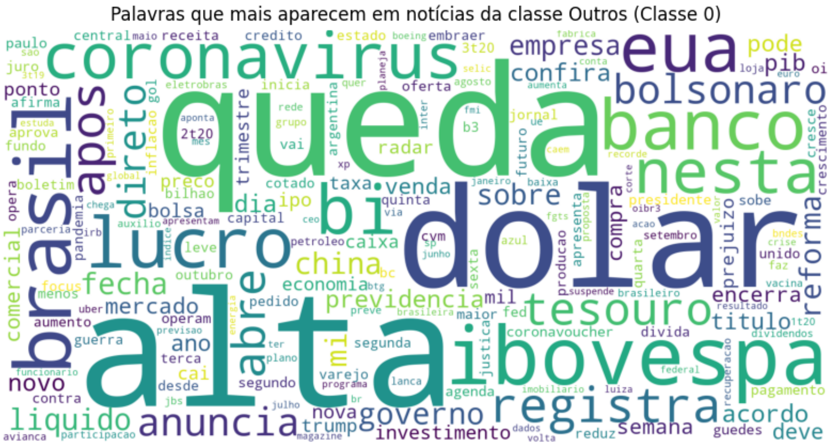


Figura 5.4: Nuvem com palavras mais recorrentes dos títulos de notícias classificados como Outros (Classe 0) (fonte: autoria própria).

Fundamentação Teórica (Seção 2) sobre *Bag of Words*, percebe-se que o *token* "atingido" é menos frequente que os demais *tokens* ao considerar os títulos do *corpus* de *treino*. Isso é constatado ao analisar seu alto valor *TF-IDF* (em torno de 0.71) em relação aos demais *tokens* (em torno de 0.50), mostrando que o *BoW TF-IDF* leva em consideração não só a frequência das palavras em um documento, mas em todo o *corpus*.

- Frase 0: *justica mg quer vale adiante <NUM> mil indenizacao cada atingido*
- Frase 1: *ministro tcu vital rego vira reu lava jato corrupcao lavagem dinheiro*

	atingido	corrupcao	dinheiro	indenizacao	justica	lavagem	ministro
0	1	0	0	1	1	0	0
1	0	1	1	0	0	1	1

Figura 5.5: Amostras com representação *BoW* tipo Contagem para Tarefa 1.

	atingido	corrupcao	dinheiro	indenizacao	justica	lavagem	ministro
0	0.710345	0.000000	0.000000	0.552057	0.436627	0.00000	0.000000
1	0.000000	0.508201	0.477835	0.000000	0.000000	0.57656	0.425423

Figura 5.6: Amostras com representação *BoW* tipo *TF-IDF* para Tarefa 1.

Otimização dos modelos (GridSearchCV)

Os hiperparametros otimizados, utilizando a função **GridSearchCV** (comentada na Seção 4.4.3), e o desempenho dos modelos no *split* de *validação* (*splits* da Tabela 4.1) estão

Tabela 5.1: Conjunto de argumentos encontrados após otimização dos modelos tradicionais da Tarefa 1.

Modelo	**Hiperparâmetro**	**Argumento para Contagem**	**Argumento para *TF-IDF***
KNN	n_neighbors	21	60
	weights	distance	distance
	metric	cosine	cosine
SVM	C	10	10
	kernel	rbf	rbf
NB	alpha	1	10
	fit_prior	True	False
LR	penalty	l2	l2
	C	1	10
	solver	liblinear	liblinear

representados na Tabela 5.1 e Tabela 5.2, respectivamente. O intervalo dos argumentos considerado para cada hiperparâmetro está na Tabela 4.5. Nota-se que, considerando a métrica acurácia, o modelo LR foi o melhor para a representação Contagem (79.62% de acurácia), enquanto o modelo *SVM* foi superior na representação *TF-IDF* (80.44% de acurácia).

Tabela 5.2: Resultados da etapa de otimização dos parâmetros dos modelos tradicionais considerando a métrica acurácia e os *splits* de treino e validação.

Modelo	**Contagem**	**TF-IDF**
KNN	77.35%	73.56%
SVM	79.59%	**80.44**%
NB	77.71%	75.14%
LR	**79.62**%	79.46%

5.2.3 Aprendizado profundo

Esta seção apresenta os principais resultados das etapas de extração de características e otimização dos modelos de *deep learning* (Subseção 4.4.4) da Tarefa 1.

Extração de Características (word2vec)

A partir do treinamento do extrator de características *word2vec*, foram gerados *embeddings* para 3001 palavras, com 300 características cada uma. Na Figura 5.7 pode ser visto como as palavras (ou *tokens*) estão organizadas no espaço vetorial criado pelo modelo de incorporação de palavras e as palavras mais similares à palavra “petr4”. A visualização foi

feita com o auxílio da classe **TSNE**[5] (implementação do método *t-distributed stochastic neighbor embedding*). O uso dessa ferramenta permitiu a redução de dimensionalidade do vetor de características de tamanho 300 serem transformados para vetores de tamanho 2, possibilitando serem representados no espaço vetorial. A partir da visualização, concluise que as palavras "petr4", "venda" e "petroleo" são muito similares, por estarem muito próximas, mostrando que frequentemente estão no mesmo contexto de uma notícia. Já na Tabela 5.3, pode ser visto as palavras mais similares em relação à palavra "petr4" (quanto próximo de um, mais similar). Percebe-se que as palavras "petrobras" e "venda" são bem similares a palavra "petr4".

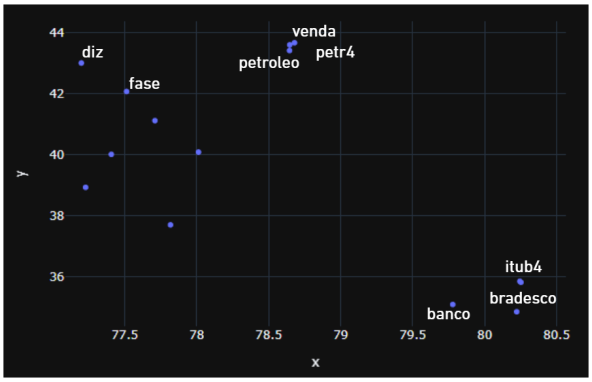


Figura 5.7: Visualização da vetorizacao word2vec da Tarefa 1.

Tabela 5.3: Palavras mais similares à "petr4", considerando apenas o *split* de treino.

Token	**Similaridade com "petr4"**
petrobras	0.9889
venda	0.9739
diz	0.9633
sal	0.9599
vinculante	0.9551
valemenos	0.9510

Otimização dos modelos (RandomSearch e EarlyStopping)

Nesta seção são apresentadas as curvas de aprendizado dos modelos *deep learning* otimizados, utilizando a função **RandomSearch**, e considerando o critério de parada antecipada, utilizando a função **EarlyStopping**, para a Tarefa 1. Ambas funções são comentadas na Subseção 4.4.4. Para isso, foi considerado o *split* de validação e a métrica acurácia para a avaliação desta etapa. Os melhores argumentos para os modelos *LSTM* e *Bi-LSTM*,

[5]https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

cujos intervalos estão na Tabela 4.6, para cada hiperparâmetro são expostos nas Tabelas 5.4 e 5.5, respectivamente. Já as curvas de aprendizado do treinamento e validação, em relação à acurácia e *loss*, do *LSTM* e *Bi-LSTM* estão ilustradas nas Figuras 5.8 e 5.9, respectivamente.

Ao analisar as curvas de acurácia e *loss* das Figuras 5.8 e 5.9, conclui-se que ambos modelos foram capazes de aprender com os dados. Isso pode ser observado ao considerar o gráfico da medida de acurácia, pois tanto a curva do treinamento quanto a de validação crescem com o passar das *epochs* (mostrando que o modelo foi acertando cada vez mais com o passar do tempo). Além disso, considerando o gráfico da medida de *loss*, as duas curvas descem com o passar das *epochs* (mostrando que o erro diminuiu tanto para o treinamento e validação com o passar do tempo), sendo mais um indício de que os modelos aprenderam. Ademais, percebe-se que o melhor modelo é o *Bi-LSTM*. Isso se justifica pelo fato da curva de validação do gráfico de acurácia estar mais próxima e com um valor maior, com o passar das *epochs*, em relação a mesma curva para o modelo *LSTM*. Em relação aos hiperparâmetros otimizados, percebe-se que muitos argumentos tiveram seu valor máximo atribuído, como é o caso da quantidade de neurônios da camada densa e *LSTM*, que tiveram valor máximo (64). Portanto, a otimização deveria ter sido feita considerando uma maior quantidade de neurônios, não apenas 16, 32 e 64. Também foi percebido que, em ambos modelos *deep learning*, a melhor função de ativação para camada densa e otimizador da rede foi a função *tanh* e o otimizador adam, respectivamente.

Tabela 5.4: Resultado dos parâmetros otimizados para *LSTM* utilizando os *splits* de treino e validação.

Hiperparâmetro	**Argumento**
Neurônios na primeira camada *LSTM*	64
Taxa de esquecimento primeiro Dropout	0.1
Neurônios na segunda camada *LSTM*	64
Neurônios na camada densa	64
Função de ativação da camada densa	tanh
Taxa de esquecimento segundo Dropout	0
Otimizador	adam

5.2.4 Avaliação dos modelos otimizados

O resultado da avaliação dos 10 modelos otimizados, processo descrito na Subseção 4.4.5, utilizando o *split* de teste (Tabela 4.1) estão nas Tabelas 5.6, 5.7 e 5.8, que contêm os melhores modelos para as representações Contagem, *TF-IDF* e *word2vec*, respectivamente. Além disso, possuem o valor médio das quatro métricas utilizadas neste trabalho (Subseção

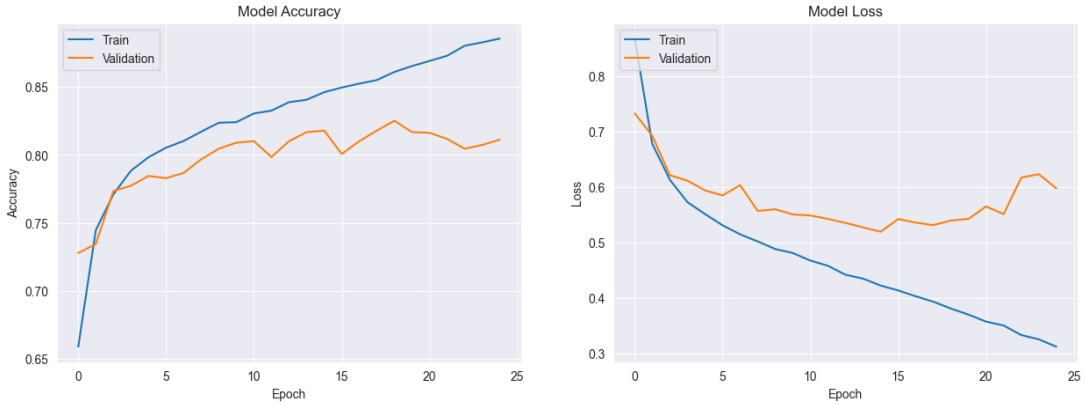


Figura 5.8: Desempenho do modelo *LSTM* em relação a loss e acurácia.

Tabela 5.5: Resultado dos parâmetros otimizados para *Bi-LSTM* utilizando os *splits* de treino e validação.

Hiperparâmetro	**Argumento**
Neurônios na primeira camada *Bi-LSTM*	16
Taxa de esquecimento primeiro Dropout	0.2
Neurônios na segunda camada *Bi-LSTM*	64
Neurônios na camada densa	64
Função de ativação da camada densa	tanh
Taxa de esquecimento segundo Dropout	0
Otimizador	adam

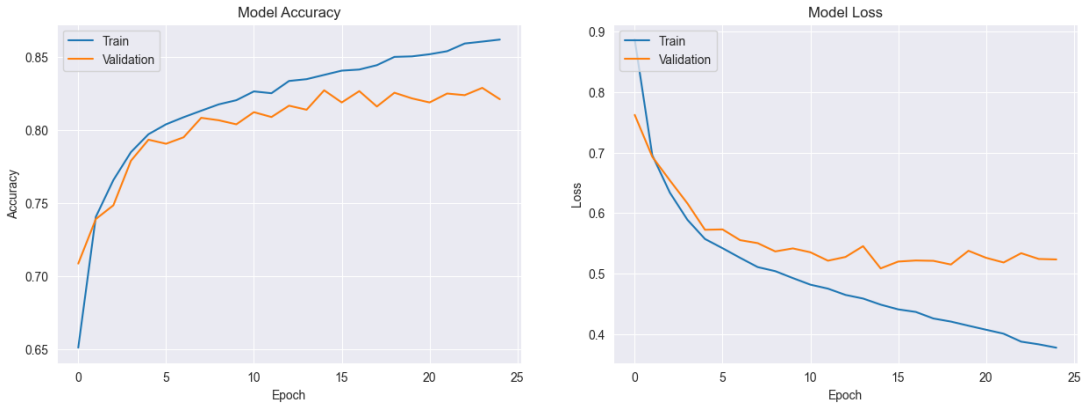


Figura 5.9: Desempenho do modelo *Bi-LSTM* em relação a loss e acurácia.

4.4.5). Conforme comentado na Subseção 4.4.5, as métricas acurácia e *F1-Score* são as métricas priorizadas para decidir o melhor modelo para cada representação.

Considerando a representação Contagem, o modelo LR foi o melhor, com 80.69% e 75.46% de acurácia e *F1-Score*, respectivamente. O modelo *SVM* também foi muito pro-

missor, com uma acurácia de 80.30% e *F1-Score* de 75.65%. Já em relação à representação *TF-IDF*, o modelo *SVM* foi o melhor, com 80.75% de acurácia e 76.31% de *F1-Score*. Diferente da representação do tipo Contagem, esse modelo foi o melhor para as duas métricas, não havendo empate. No caso dos modelos de *deep learning*, o *LSTM* também foi melhor que o Bi-LSTM nas duas métricas, com 83.07% de acurácia e com 81.19% de *F1-Score*. Portanto, *LSTM* foi considerado o melhor para a representação *word2vec*. Dessa forma, percebe-se que os melhores modelos para as representações Contagem, *TF-IDF* e *word2vec* são, respectivamente LR, *SVM* e *LSTM*. Ademais, considerando esses três melhores modelos, o modelo *LSTM* obteve a melhor acurácia e *F1-Score*. Portanto, o melhor modelo para a Tarefa 1, considerando as predições no *split* de teste, é o *LSTM* com a representação *word2vec*.

Tabela 5.6: Avaliação dos modelos do tipo Contagem da Tarefa 1.

Modelo	**Acurácia (%)**	**Precisão (%)**	**Revocação (%)**	***F1-Score* (%)**
KNN	78.86	77.43	72.94	74.43
SVM	80.30	77.26	74.79	**75.65**
NB	79.86	76.19	74.79	75.40
LR	**80.69**	77.83	74.42	75.46

Tabela 5.7: Avaliação dos modelos do tipo *TF-IDF* da Tarefa 1.

Modelo	**Acurácia (%)**	**Precisão (%)**	**Revocação (%)**	***F1-Score*(%)**
KNN	75.36	75.46	68.99	71.07
SVM	**80.74**	77.74	75.52	**76.31**
NB	76.69	74.17	70.79	72.08
LR	80.63	76.93	75.09	75.71

Tabela 5.8: Avaliação dos modelos do tipo word2vec da Tarefa 1.

Modelo	**Acurácia (%)**	**Precisão (%)**	**Revocação (%)**	***F1-Score* (%)**
LSTM	**83.07**	83.54	79.82	**81.19**
Bi-LSTM	82.13	82.06	78.77	79.97

5.2.5 Análise de resultados do melhor modelo da Tarefa 1

Para analisar os resultados do melhor modelo, decidido a partir dos resultados da Seção 5.2.4, da Tarefa 1 (*LSTM* em conjunto com a representação *word2vec*), é apresentado o gráfico de barras (Figura 5.10) das quatro métricas de avaliação utilizadas neste trabalho (Subseção 4.4.5), a matriz de confusão (Figura 5.11) contendo a quantidade de acertos

para cada classe e uma tabela contendo dez classificações corretas e incorretas feitas por esse modelo (Tabela 5.9).

Ao observar a Tabela 5.8, percebe-se que o modelo *LSTM* possui um valor médio de 79.82% de revocação e 81.19% de *F1-Score*. Porém, ao analisar gráfico da Figura 5.10, percebe-se que a classe Itaú obteve uma revocação de quase 60%, além de possuir um *F1-Score* em torno de 70%. Portanto, esses resultados mostram que analisar apenas os valores médios das quatro métricas pode esconder o desempenho do modelo na predição de uma classe específica, como a classe Itaú.

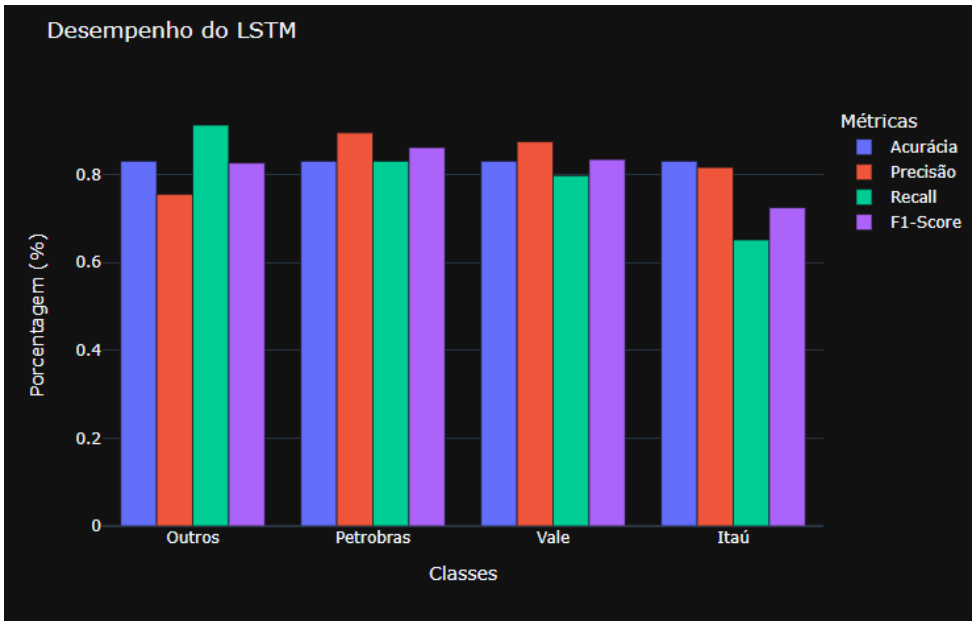


Figura 5.10: Gráfico de métricas do *LSTM*.

Já ao analisar a matriz de confusão (Figura 5.11), percebe-se que, observando a diagonal principal, o modelo acertou 564, 280, 129 e 524 nas classes Petrobras, Vale, Itaú e Outros, respectivamente. A alta quantidade de acertos confirma o desempenho apresentado pela Tabela 5.8.

Observando a Tabela 5.9, que possui as dez classificações realizadas pelo modelo, era esperado que o modelo acertasse a amostra "6", devido ao fato do título de notícia conter as palavras "itau" e "carteira", palavras bastante frequentes na nuvem de palavras desta classe (Figura 5.3), porém, isso não ocorreu. Apesar disso, o modelo teve êxito em classificar a amostra "4", apesar de não conter a palavra mais frequente desta categoria (classe "petrobras"). Portanto, considerando essa amostra de classificações e os resultados da avaliação do melhor modelo dessa tarefa, conclui-se que os títulos do *dataset* desta tarefa possuem informações relevantes para possibilitar que o modelo aprenda a categorizar os títulos de notícias nas classes Petrobras, Itaú, Vale e Outros.

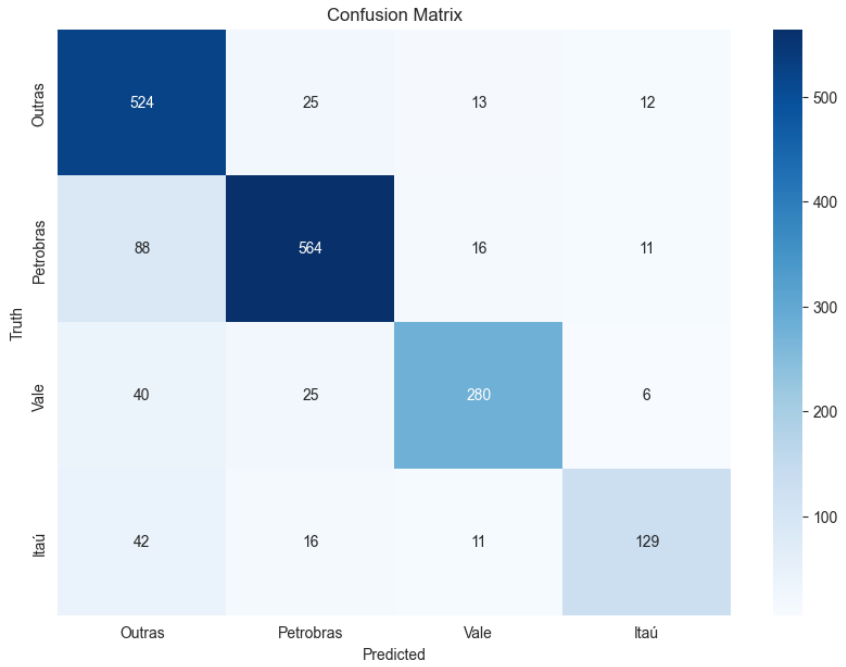


Figura 5.11: Matriz de confusão do *LSTM*.

Tabela 5.9: Amostra do resultado das predições do modelo *LSTM*.

	title	y_real	y_pred	classificação
0	so acao sobrevive carteira semanal xp veja	2	1	Incorreta
1	ivan monteiro ainda definiu aceita comandar bb <NUM> diz estadao	1	0	Incorreta
2	petrobras lista privatizacoes primeiro mandato diz guedes	1	1	Correta
3	magazine luiza mglu3 petrobras petr4 agitam mercado financeiro	1	1	Correta
4	anp registra vazamento gas oleo plataforma trident energy campos	1	1	Correta
5	copom prega cautela indica juro estavel proximos meses	0	0	Correta
6	esperamos carteira credito itau estabilize <NUM> semestre diz bracher	3	1	Incorreta
7	itau itub4 vence licitacao mg gerir folha pagamentos estado	3	3	Correta
8	oi anuncia eurico teles saira presidencia <NUM> janeiro	0	0	Correta
9	bolsonaro cobra petrobras petr4 sobre reducao preco apos queda petroleo	1	1	Correta

5.3 Resultados experimentais da Tarefa 2

Esta seção é responsável por mostrar os principais resultados obtidos da normalização textual, extração de características, otimização dos dos hiperparâmetros e avaliação dos modelos a partir do *dataset* criado para a Tarefa 2.

5.3.1 Normalização textual

De forma similar à Tarefa 1, também foi gerado nuvem de palavras para cada classe desta tarefa após a normalização de textos do *dataset*. Analisando as nuvens da Figura 5.12, Figura 5.13 e Figura 5.14, foi observado como as palavras mais frequentes de cada classe as palavras "venda" e "alta", mostrando que, diferente das nuvens da Tarefa 2, os títulos desse *dataset* não possuem palavras que diferenciam uma classe da outra. Portanto, espera-se que os modelos preditivos tenham dificuldade em aprender a diferenciar cada uma dessas classes durante o treinamento. Consequentemente, também é esperado que os resultados da etapa de avaliação desses modelos não sejam satisfatórios.

Figura 5.12: Nuvem com palavras mais recorrentes dos títulos de notícias classificados como Desceu (Classe 0) (fonte: autoria própria).

5.3.2 Aprendizado tradicional

Esta seção apresenta os principais resultados das etapas de extração de características e otimização dos modelos tradicionais (Seção 4.4.3) da Tarefa 2.

Extração de Características (Contagem e *TF-IDF*)

Para mostrar os resultados dessa etapa, optou-se por selecionar duas amostras do conjunto de treino (*Frase0* e *Frase1*) para mostrar seu mapeamento no espaço vetorial criado

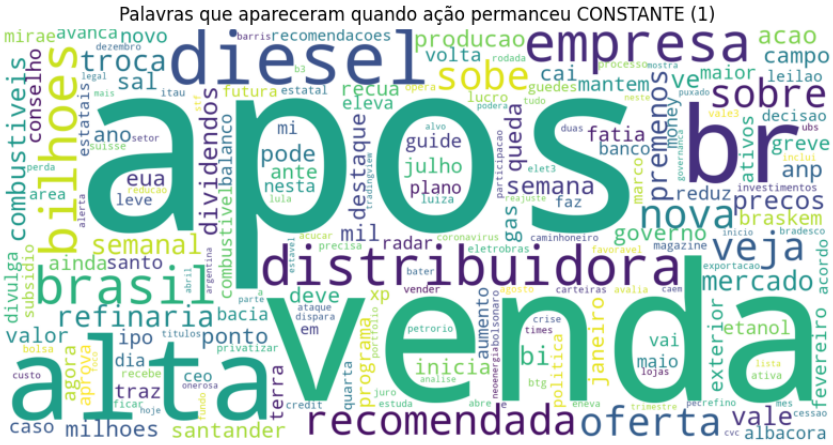


Figura 5.13: Nuvem com palavras mais recorrentes dos títulos de notícias classificados como Constante (Classe 1) (fonte: autoria própria).

Figura 5.14: Nuvem com palavras mais recorrentes dos títulos de notícias classificados como Subiu (Classe 2) (fonte: autoria própria).

pelas representações numéricas Contagem (Equação 2.1) e *TF-IDF* (Equação 2.2), que podem ser vistos na Figura 5.15 e Figura 5.16, respectivamente. Conforme explicado na Subseção 2.3.2, a primeira representação irá criar o espaço vetorial baseado na frequência de cada palavra do documento, já o *TF-IDF*, se baseia não só na frequência de cada palavra no documento, mas na frequência dessa em todo o *corpus*. Isso pode ser percebido ao se analisar as palavras "petrobras" nos dois resultados. Observando o *BoW* de tipo Contagem, o fato de aparecer a palavra "petrobras" já resulta no valor 1 nesse atributo, enquanto no tipo *TF-IDF*, esse valor é 0.135588, indicando que apesar desta palavra ocorrer nesse documento, ela está muito frequente nos demais documentos do *corpus*, sendo penalizada. Essa característica do *TF-IDF* é importante para auxiliar os modelos preditivos na discriminação das notícias para cada rótulo. Os resultados da extração de

características do tipo Contagem e *TF-IDF* de duas amostras do treino:

- Frase 0: *petrobras petr4 sindicato aprova acordo coletivo trabalho*
- Frase 1: *ibovespa patina manter <NUM> mil pontos embraer dispara*

	acordo	aprova	dispara	embraer	ibovespa	manter	petrobras	pontos	sindicato	trabalho
0	1	1	0	0	0	0	1	0	1	1
1	0	0	1	1	1	1	0	1	0	0

Figura 5.15: Amostras vetorizadas com *BoW* tipo Contagem para Tarefa 2.

	acordo	aprova	dispara	embraer	ibovespa	manter	petrobras	pontos	sindicato	trabalho
0	0.409233	0.424712	0.000000	0.000000	0.000000	0.000000	0.135588	0.000000	0.562923	0.562923
1	0.000000	0.300000	0.435464	0.550936	0.191271	0.586971	0.000000	0.354571	0.000000	0.000000

Figura 5.16: Amostras vetorizadas com *BoW* tipo *TF-IDF* para Tarefa 2.

Otimização dos modelos (GridSearchCV)

Os hiperparametros otimizados, utilizando a função **GridSearchCV** (comentada na Seção 4.4.3), e o desempenho dos modelos no *split* de *validação* (*splits* da Tabela 4.3) da Tarefa 2 estão representados na Tabela 5.10 e Tabela 5.11, respectivamente. O intervalo dos argumentos considerado para cada hiperparâmetro está na Tabela 4.5. Percebe-se que, considerando a métrica acurácia, o modelo LR foi o melhor para a representação Contagem (57.56% de acurácia), enquanto o modelo *KNN* foi superior na representação *TF-IDF* (57.71% de acurácia).

5.3.3 Aprendizado profundo

Esta seção apresenta os principais resultados das etapas de extração de características e otimização dos modelos *deep learning* (Seção 4.4.4) da Tarefa 2.

Extração de Características (word2vec)

Após o treinamento do modelo *word2vec*, foram gerados *embeddings* para 574 palavras, com 300 características cada uma. A quantidade de palavras com *embeddings* foi bem menor em relação a quantidade da Tarefa 1 por conta da menor quantidade de amostras para treino. Na Figura 5.17 pode ser visto como as palavras estão organizadas no espaço vetorial criado pelo modelo de incorporação de palavras e também as mais similares à palavra “petr4”. A partir da visualização, conclui-se que as palavras “petr4” e “acoes”

Tabela 5.10: Conjunto de argumentos encontrados após otimização dos modelos tradicionais da Tarefa 2.

Modelo	**Hiperparâmetro**	**Argumento para Contagem**	**Argumento para *TF-IDF***
KNN	n_neighbors	23	23
	weights	distance	distance
	metric	cosine	cosine
SVM	C	1	1
	kernel	rbf	rbf
NB	alpha	10	1
	fit_prior	True	True
LR	penalty	l2	l1
	C	0.1	1
	solver	sag	liblinear

Tabela 5.11: Resultados da etapa de otimização dos parâmetros dos modelos tradicionais considerando a métrica Acurácia e os *splits* de treino e validação

Modelo	**Contagem**	***TF-IDF***
KNN	55.97%	**57.71**%
SVM	57.45%	57.01%
NB	57.39%	55.97%
LR	**57.56**%	57.06%

são muito similares, mostrando que frequentemente estão no mesmo contexto de uma notícia. Já na Tabela 5.12, pode ser visto as palavras mais similares em relação à palavra "petr4". Percebe-se que as palavras "mercado" e "elet3" são bem similares a essa palavra, apesar de não estarem destacadas no espaço vetorial de duas dimensões.

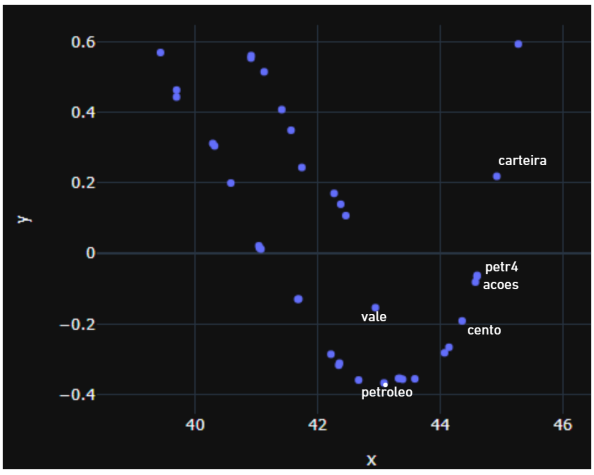


Figura 5.17: Visualização da vetorização *word2vec* da tarefa 2.

Tabela 5.12: Palavras mais similares à “petr4”

Token	**Similaridade com “petr4”**
mercado	0.8972
elet3	0.8876
financeiro	0.8797
eletrobras	0.8726
atencao	0.8672
chamam	0.8558

Otimização dos modelos (RandomSearch e EarlyStopping)

Nesta seção são apresentadas as curvas de aprendizado dos modelos *deep learning* otimizados, utilizando a função **RandomSearch**, e considerando o critério de parada antecipada, utilizando a função **EarlyStopping**, para a Tarefa 2. Ambas funções são comentadas na Subseção 4.4.4. Os melhores argumentos para os modelos *LSTM* e *Bi-LSTM*, cujos intervalos estão na Tabela 4.6, são expostos nas Tabelas 5.4 e 5.5, respectivamente. As curvas de desempenho do treinamento e validação estão ilustradas na Figura 5.8 e Figura 5.9.

Ao analisar as curvas de acurácia e *loss* das Figuras 5.18 e 5.19, percebe-se que o modelo decorou a partir dos dados em vez de aprender. Isso é demonstrado ao ver a curva de treinamento do gráfico de acurácia aumentar com o passar das *epochs* mas a curva de validação não obter o mesmo comportamento, permanecendo constante até a época 7 e depois caindo. Comportamento também pode ser visto no gráfico da *loss*, em que a curva de treinamento cai com o passar das épocas (mostrando que o erro do treinamento está diminuindo com o passar do tempo), enquanto a curva de validação possui um comportamento de subida (mostrando que o erro da validação está aumentando com o passar do tempo).

Tabela 5.13: Resultado dos parâmetros otimizados para *LSTM*.

Hiperparâmetro	**Argumento**
Neurônios na primeira camada *LSTM*	32
Taxa de esquecimento primeiro Dropout	0
Neurônios na segunda camada *LSTM*	32
Neurônios na camada densa	32
Função de ativação da camada densa	tanh
Taxa de esquecimento segundo Dropout	0.1
Otimizador	adam

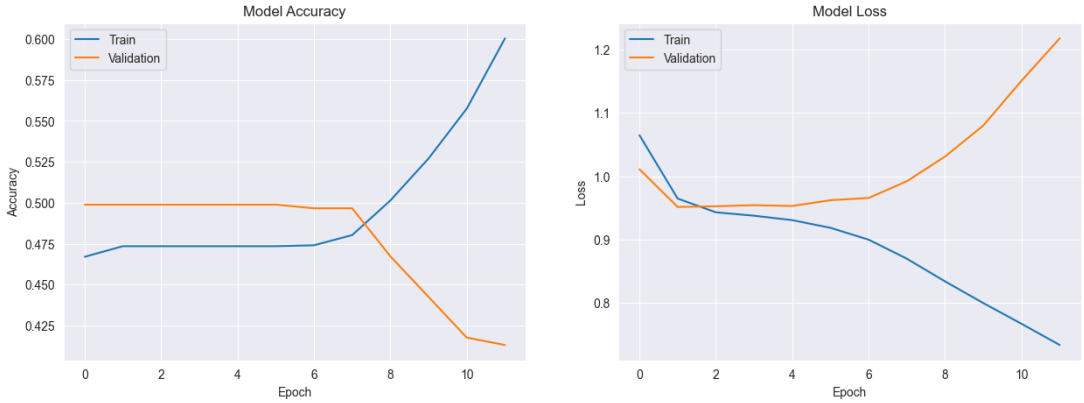


Figura 5.18: Desempenho do modelo *LSTM* em relação a loss e acurácia.

Tabela 5.14: Resultado dos parâmetros otimizados para *Bi-LSTM*.

Hiperparâmetro	**Argumento**
Neurônios na primeira camada *Bi-LSTM*	16
Taxa de esquecimento primeiro Dropout	0
Neurônios na segunda camada *Bi-LSTM*	16
Neurônios na camada densa	16
Função de ativação da camada densa	tanh
Taxa de esquecimento segundo Dropout	0.1
Otimizador	adam

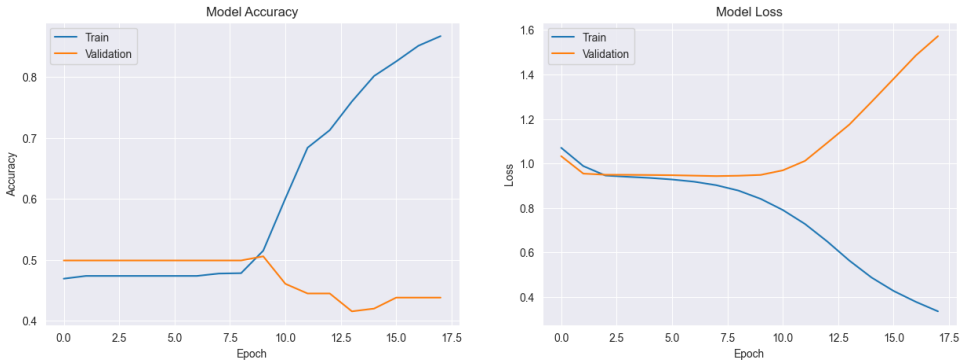


Figura 5.19: Desempenho do modelo *Bi-LSTM* em relação a loss e acurácia.

5.3.4 Avaliação dos modelos otimizados

O resultado da avaliação dos 10 modelos otimizados utilizando o *split* de teste estão nas Tabelas 5.15, 5.16 e 5.17, que contém os melhores modelos para as representações Contagem, *TF-IDF* e *word2vec*, respectivamente. Além disso possuem o valor médio das quatro métricas utilizadas neste trabalho. Porém, as métricas acurácia e *F1-Scores* são priorizadas para decidir o melhor modelo (conforme comentado na Seção 4.4.5), de forma

similar ao que foi feito na Tarefa 1.

Considerando a representação Contagem, o modelo *NB*, com acurácia de 50.45% e *F1-Scores* de 33.27%, e o modelo *KNN*, com acurácia de 50.23% de acurácia e 33.84% de *F1-Scores*, atingiram bons resultados em relação aos demais. Já ao considerar a representação *TF-IDF*, o modelo *SVM* obteve os melhores resultados para as duas métricas, não havendo empate. O modelo atingiu 50.68% de acurácia e 34.01% de *F1-Scores*. Considerando os modelos de *deep learning* (representação *word2vec*), também não houve empate, com o modelo *Bi-LSTM* sendo o melhor para as duas métricas, com 44.81% de acurácia e 35.78% de *F1-Scores*. Assim, os melhores modelos para as representações Contagem, *TF-IDF* e *word2vec* foram, respectivamente, *NB*, *SVM* e *Bi-LSTM*. Observando a métrica acurácia, o modelo *SVM* foi o melhor, com 50.68%. Porém, ao observar a métrica *F1-Scores*, o modelo *Bi-LSTM* foi melhor, com 35.78%. Mas, como a acurácia do modelo de *deep learning* foi muito menor do que a acurácia do *SVM*, decidiu-se que o *SVM* com representação *TF-IDF* foi o melhor modelo, considerando as predições no *split* de teste, para a Tarefa 2.

Tabela 5.15: Avaliação dos modelos do tipo Contagem da Tarefa 2.

Modelo	**Acurácia (%)**	**Precisão (%)**	**Revocação (%)**	***F1-Score* (%)**
KNN	50.23	33.79	35.81	**33.84**
SVM	49.77	33.08	35.43	33.09
NB	**50.45**	33.79	35.85	33.27
LR	48.65	32.25	34.69	32.63

Tabela 5.16: Avaliação dos modelos do tipo *TF-IDF* da Tarefa 2.

Modelo	**Acurácia (%)**	**Precisão (%)**	**Revocação (%)**	***F1-Score* (%)**
KNN	49.55	33.28	35.42	33.72
SVM	**50.68**	33.79	36.14	**34.01**
NB	48.42	32.12	34.60	32.73
LR	49.77	33.37	35.12	31.30

Tabela 5.17: Avaliação dos modelos do tipo word2vec da Tarefa 2.

Modelo	**Acurácia (%)**	**Precisão (%)**	**Revocação (%)**	***F1-Score* (%)**
LSTM	43.69	33.59	33.61	33.27
Bi-LSTM	**44.81**	36.20	35.66	**35.78**

5.3.5 Análise de resultados do melhor modelo da Tarefa 2

Para analisar os resultados do melhor modelo, decidido a partir dos resultados da Seção 5.3.4, da Tarefa 2 (*SVM* em conjunto com a representação *TF-IDF*), é apresentado o

gráfico de barras (Figura 5.20) das quatro métricas de avaliação utilizadas neste trabalho (Subseção 4.4.5), a matriz de confusão (Figura 5.21) contendo a quantidade de acertos para cada classe e uma tabela contendo dez classificações corretas e incorretas feitas por esse modelo (Tabela 5.18).

Ao observar a Tabela 5.16, percebe-se que o modelo *SVM* possui um valor médio de 33.79%, 36.14% e 34.01% para as métricas precisão, revocação e *F1-Score*, respectivamente. Porém, ao analisar gráfico da Figura 5.20, é observado que a classe Constante obteve um valor nulo nas três métricas citadas. Ou seja, o modelo não conseguiu acertar nenhuma classe Constante. Portanto, se confirma novamente a importância de, além de analisar o valor médio das métricas de desempenho dos modelos, analisar também o valor das métricas para cada classe.

Ao observar a diagonal principal da matriz de confusão da Figura 5.21, se confirma o fenômeno do modelo não acertar nenhuma classe do tipo Constante, pois a célula central (que indica quanto acertos foram feitos para essa classe) possui valor nulo. Apesar disso, o modelo acertou corretamente 153 amostras na classe Desceu e 72 amostras na classe Subiu.

Observando a Tabela 5.18, que possui as dez classificações realizadas pelo melhor modelo da Tarefa 2 e os baixos valores médios para as quatro métricas da Tabela 5.16, percebe-se que é difícil prever a movimentação da ação da Petrobras apenas observando o título de uma notícia, pois muitas vezes essas notícias não possuem nenhuma informação que afete a movimentação do seu preço (amostras “0”, “6”, “7” e “8”, por exemplo). Observando mais detalhadamente a amostra “0”, o modelo classificou corretamente que a ação vai ter seu preço diminuído no dia dessa notícia, mas o título não contém nenhuma palavra que dá um indício de que a ação, de fato, vai cair (não há palavras como “queda”, “cair”, ou alguma informação que prejudique a Petrobras). Portanto, esse fenômeno justifica, em parte, a dificuldade do modelo em aprender a prever a movimentação da ação da PETR4 utilizando apenas as informações do título do *dataset* desta tarefa.

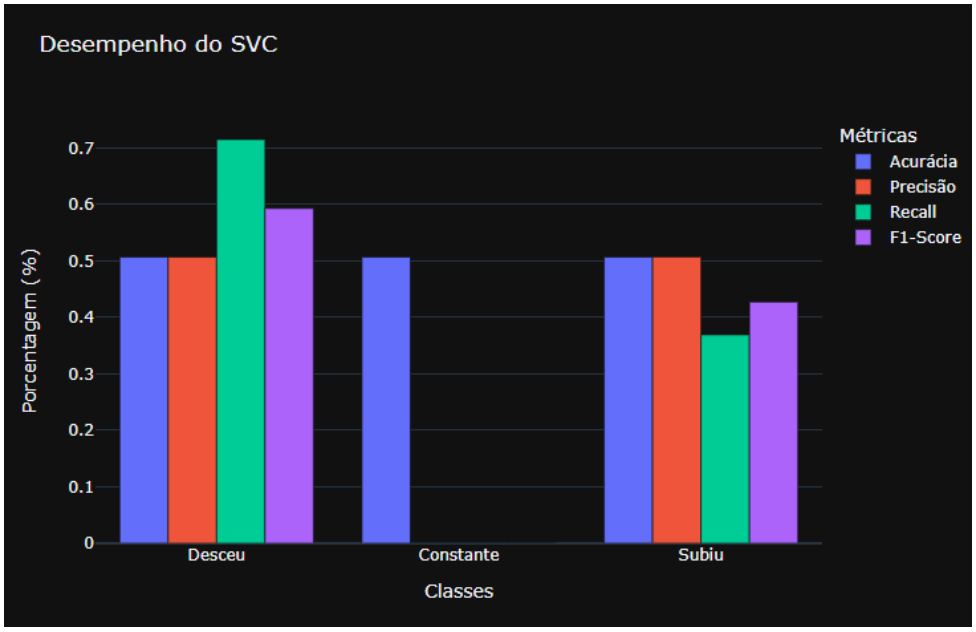


Figura 5.20: Gráfico de métricas do *SVM*.

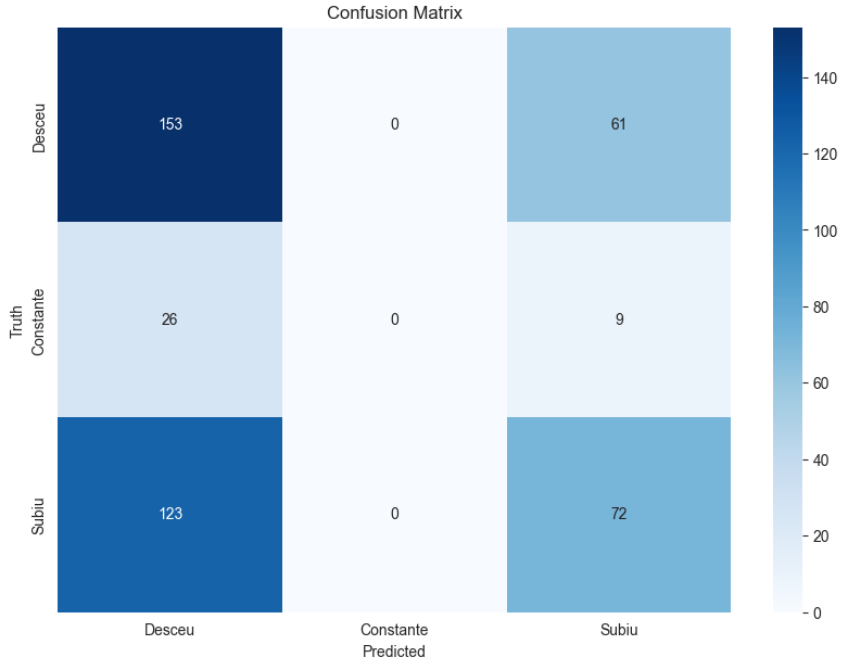


Figura 5.21: Matriz de confusão do *SVM*.

Tabela 5.18: Amostra do resultado das predições do modelo *SVM*.

	title	y_real	y_pred	classificação
0	b3 divulga nova previa ibovespa quadrimestre veja mudancas	0	0	Correta
1	ibovespa vale vale3 sabesp sbsp3 lideram altas inicio pregao	2	0	Incorreta
2	petrobras petr4 meirelles defende divisao estatal tres quatro empresas privatizacao	2	0	Incorreta
3	diesel aproxima <NUM> litro postos aponta ticket log	2	0	Incorreta
4	acao petrobras desconto <NUM> cento oportunidade compra	0	0	Correta
5	ibovespa recuperacao leve investidor segue atento china	2	2	Correta
6	petrobras lanca iniciativa primeira infancia	2	2	Correta
7	premenos market money times tudo precisa saber agora	1	0	Incorreta
8	petrobras prorroga inscricoes financiamento projetos culturais	0	0	Correta
9	ibovespa afunda <NUM> cento apos ataque russo ucrania	0	0	Correta

Capítulo 6

Discussão dos Resultados

Este capítulo tem o objetivo de analisar os principais resultados da avaliação dos modelos tradicionais e de *deep learning* para as Tarefas 1 e 2. Além disso, é responsável pela análise dos melhores modelos para cada tipo de representação numérica, representados nas Tabelas 6.1 e 6.2, respectivamente. Além disso, essas tabelas consideram as quatro métricas Acurácia, Precisão, Revocação e *F1-Score* apresentadas na Seção 4.4.5, representadas pelas abreviações Acc., Prec., Rev., e F1., respectivamente.

A Tarefa 1 gerou resultados satisfatórios, com acurácia acima de 80% para as três representações de extração de características, conforme apresentado na Tabela 6.1. Considerando as três representações e as métricas acurácia e *F1-Score*, os melhores modelos para as representações Contagem, TF-IDF e word2vec foram LR, *SVM* e *LSTM*, respectivamente. Desses, o que obteve o melhor resultado foi o modelo *LSTM* com representação *word2vec*, com acurácia de 83.07% e *F1-Score* de 81.19%. Em relação a Tarefa 2, ao observar os resultados das métricas da Tabela 6.2, conclui-se que os modelos não tiveram sucesso em prever o movimento do preço da ação PETR4, da Petrobras. Apesar dos resultados não satisfatórios, o melhor modelo para essa tarefa foi o *SVM* junto da representação TF-IDF, com acurácia e *F1-Score* de 50.68% e 34.01%, respectivamente. Em relação a esse modelo, foi percebido pela sua matriz de confusão que ele não conseguiu acertar nenhuma amostra como pertencendo à classe Constante. Acredita-se que isso se deu pelo desbalanceamento desta classe com as demais. Um aumento da porcentagem do *threshold*, definido na Subseção 4.4.1, poderia ajudar a melhorar esse desbalanceamento.

Tabela 6.1: Melhores resultados da Tarefa 1.

Modelo	**Representação**	**Acc. (%)**	**Prec. (%)**	**Rev. (%)**	**F1. (%)**
LR	Contagem	80.69	77.83	74.42	75.46
SVM	*TF-IDF*	80.74	77.74	75.52	76.31
LSTM	*Word2vec*	**83.07**	83.54	79.82	**81.19**

Tabela 6.2: Melhores resultados da Tarefa 2.

Modelo	**Representação**	**Acc. (%)**	**Prec. (%)**	**Rev. (%)**	**F1. (%)**
NB	Contagem	50.45	33.79	35.85	33.27
SVM	*TF-IDF*	**50.68**	33.79	36.14	34.01
Bi-LSTM	*Word2vec*	44.81	36.20	35.66	**35.78**

A partir da análise dos resultados das duas tarefas, observa-se que a Tarefa 1 obteve êxito em categorizar notícias nas classes Petrobras, Itaú, Vale e Outras, com a avaliação do seu melhor modelo adquirindo aproximadamente 80% em todas as métricas. Além disso, os resultados desta tarefa mostraram que é possível fazer a categorização de notícias em classes relacionadas à empresas, abrindo precedente para que mais classes deste escopo possam vir a ser adicionadas para sua predição. Porém, no caso da Tarefa 2, a predição da movimentação do preço da PETR4 não foi bem sucedida, com a maior parte das métricas de avaliação do melhor modelo sendo abaixo de 50%, sendo apenas a métrica acurácia acima dessa porcentagem. Acredita-se que os resultados não satisfatórios da Tarefa 2 se deram por dois principais motivos: (1) a grande dificuldade para se prever a movimentação do preços de ações do mercado financeiro, por conta dos diversos fatores que a influenciam diariamente e (2) a escolha de uma ação que costuma sofrer mais variações que as demais, por ser influenciada por decisões políticas, preço dos barris de petróleo, valor do dólar, entre outros fatores.

Capítulo 7

Conclusão e Trabalhos Futuros

O presente trabalho propôs um método para a exploração de duas tarefas: a classificação de títulos de notícias do mercado financeiro nas classes Petrobras, Itaú, Vale e Outras; e a de prever o comportamento do preço da ação da Petrobras (PETR4). Ambas tarefas utilizam de conceitos de aprendizagem de máquina e processamento de linguagem natural para produzir seus resultados experimentais.

Na Tarefa 1, os resultados foram satisfatórios, com o melhor modelo para a tarefa sendo o modelo *LSTM* junto da representação *word2vec*; e atingindo 83.07 e 81.19 de acurácia e *F1-Scores*, respectivamente. Além disso, o modelo foi capaz não só de classificar corretamente títulos que possuem o nome das empresas no texto, como também títulos que não continham, provando que o modelo pode superar mecanismos de busca por palavras-chave, em que só recupera títulos com aquela palavra específica. Já na Tarefa 2, os resultados não foram satisfatórios, sendo o modelo *SVM* junto da representação TF-IDF o que apresentou melhores resultados, com 50.68% e 34.01% de acurácia e *F1-Scores*, respectivamente. Acredita-se que isso se deve à grande dificuldade em prever o movimento do preço de ações do mercado financeiro [6], e também pela escolha de uma ação que sofre influências de diversos fatores externos, como é o caso da PETR4, ação da Petrobras.

Em relação aos trabalhos futuros para experimentos relacionados classificação de notícias, podem ser feitos experimentos na base de dados , informada na Seção 4.1, com o objetivo de aumentar a quantidade de classes a serem preditas, pois apenas foram usadas: Petrobras, Vale, Itau e Outros. Além disso, pode ser interessante transformar essa tarefa em uma classificação *multi-label*, de forma similar ao que foi proposto por Nadeem *et al.* [28]. Um exemplo de classificação *multi-label* considerando as classes da Tarefa 1 seria o mesmo título de notícia podendo ser classificado como pertencendo à Petrobras e Vale, tornando a classificação mais fidedigna, pois é muito difícil uma notícia estar relacionada a uma única entidade ou empresa.

Considerando os trabalhos futuros de experimentos ligados a predição do preço de

ações do mercado financeiro brasileiro, sugere-se, além de utilizar análise fundamental, também a análise técnica para a predição. Ademais, experimentos para variar a forma com que o atributo *profit* é criado (variar o *threshold* usado para criar os rótulos, por exemplo) são interessantes com objetivo de tentar melhorar os resultados. Além disso, é interessante aplicar esse mesmo método em uma ação que sofre menos variações do que a PETR4, como ações da Ambev ou Vale, por exemplo. Além disso, é interessante abordar em trabalhos futuros os métodos de normalização de textos stemização e lematização, por já ter sido provado na literatura que possibilitam melhora nos resultados [17].

Referências

[1] Nobre, Renato A., Khalil C. do Nascimento, Patricia A. Vargas, Alan Demétrius Baria Valejo, Gustavo Pessin, Leandro A. Villas e Geraldo P. Rocha Filho: *Aurora: an autonomous agent-oriented hybrid trading service*. Em *Neural Computing and Applications*, volume 34, páginas 2217–2232, 2022. ix, 2, 5, 11, 14, 21, 23, 25

[2] Sousa, Alexandre Santana: *Análise comparativa de redes neurais convolucionais para a detecção de câncer de pulmão em tomografias computadorizadas*, 2023. ix, 15

[3] Sri Sravya, Vukyam, Sachin Kumar S e K P Soman: *Text categorization of telugu news headlines*. Em *2022 2nd International Conference on Intelligent Technologies (CONIT)*, páginas 1–6, 2022. 2, 18, 22, 23

[4] Zhang, Tianyu e Fucheng You: *Research on short text classification based on textcnn*. Em *Journal of Physics: Conference Series*, volume 1757. IOP Publishing, 2021. 2, 18, 23

[5] Schmitz, Matheus, Roger Immich, Gustavo Pessin e Geraldo Pereira Rocha Filho: *Towards the categorization of brazilian financial market headlines*. Em *IEEE Latin America Transactions*, volume 20, páginas 344–351, 2022. 2, 3, 11, 17, 22, 23, 24

[6] Nti, Isaac Kofi, Adebayo Felix Adekoya e Benjamin Asubam Weyori: *A systematic review of fundamental and technical analysis of stock market predictions*. Em *Artificial Intelligence Review*, volume 53, páginas 3007–3057, Apr 2020. 2, 3, 5, 61

[7] Chung, Hyejung e Kyung shik Shin: *Genetic algorithm-optimized long short-term memory network for stock market prediction*. Em *Sustainability*, volume 10, página 3765, outubro 2018. 2, 19, 23

[8] Nabipour, Mojtaba, Pooyan Nayyeri, Hamed Jabani, Shahab Band e Amir Mosavi: *Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis*. Em *IEEE Access*, volume 8, páginas 1–1, 2020. 2, 19, 23

[9] Sousa, Matheus Gomes, Kenzo Sakiyama, Lucas de Souza Rodrigues, Pedro Henrique Moraes, Eraldo Rezende Fernandes e Edson Takashi Matsubara: *Bert for stock market sentiment analysis*. Em *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, páginas 1597–1601, 2019. 2, 18, 19, 20, 23

[10] Kalra, Sneh e Jay Shankar Prasad: *Efficacy of news sentiment for stock market prediction*. Em *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, páginas 491–496, 2019. 2, 6, 20, 22, 23

[11] O. Carosia, Arthur E. de, Guilherme P. Coelho e Ana E. A. da Silva: *The influence of tweets and news on the brazilian stock market through sentiment analysis*. Em *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*, WebMedia '19, página 385–392, New York, NY, USA, 2019. Association for Computing Machinery. 2, 6, 20, 23

[12] Ko, Ching Ru e Hsien Tsung Chang: *Lstm-based sentiment analysis for stock price forecast*. Em *PeerJ Computer Science*, volume 7, página e408, 2021. 2, 6, 21, 23

[13] Alaparthi, Shivaji e Manit Mishra: *Bidirectional encoder representations from transformers: A sentiment analysis odyssey*, 2020. 2, 18, 23

[14] Nayak, Deepak e Bharath Bolla: *Efficient Deep Learning Methods for Sarcasm Detection of News Headlines*, páginas 371–382. fevereiro 2022. 3, 18, 23

[15] Hobson Lane, Hannes Hapke e Cole Howard: *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python, First Edition*. Manning; First Edition, 2019. 7, 8

[16] García, Salvador, Julián Luengo e Francisco Herrera: *Data Preprocessing in Data Mining*. Springer International Publishing, 2015. 7

[17] Jurafsky, Daniel e James H. Martin: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Third Edition*. Prentice Hall PTRUpper Saddle River, NJ, United States, 2023. 8, 9, 10, 62

[18] Kleene, Stephen Cole: *Mathematical Logic*. Dover Publications, 2002. 8

[19] Eisenstein, Jacob: *Introduction to Natural Language Processing*. The MIT Press; Illustrated edition (October 1, 2019). 9

[20] Mitchell, Tom M.: *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997. 11

[21] Bramer, Max: *Principles of Data Mining*. Springer Publishing Company, Incorporated, 2nd edição, 2013. 12

[22] Cortes, Corinna e Vladimir Vapnik: *Support-vector networks*. Em *Machine Learning*, volume 20, páginas 273–297, Sep 1995. 12

[23] Maron, M. E.: *Automatic indexing: An experimental inquiry*. Em *J. ACM*, volume 8, página 404–417, New York, NY, USA, 1961. Association for Computing Machinery. 13

[24] Géron, Aurélien: *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2017. 13

[25] Hochreiter, Sepp e Jürgen Schmidhuber: *Long short-term memory*. Em *Neural computation*, volume 9, páginas 1735–80, dezembro 1997. 13

[26] Zhang, Shu, Dequan Zheng, Xinchen Hu e Ming Yang: *Bidirectional long short-term memory networks for relation classification.* Em *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, páginas 73–78, Shanghai, China, outubro 2015. 14

[27] Medeiros, Murilo Cerqueira: *Metodologias para análise de sentimentos de tweets sobre o mercado financeiro, universidade de brasília, monografia*, 2019. 30

[28] Nadeem, Muhammad Imran, Kanwal Ahmed, Dun Li, Zhiyun Zheng, Hafsa Naheed, Abdullah Y. Muaad, Abdulrahman Alqarafi e Hala Abdel Hameed: *Sho-cnn: A metaheuristic optimization of a convolutional neural network for multi-label news classification.* Em *Electronics*, volume 12, 2023. 61