Universidade Federal
do Rio de Janeiro
Escola Politécnica

ANÁLISE DE NOTÍCIAS DO MERCADO FINANCEIRO UTILIZANDO PROCESSAMENTO DE LINGUAGEM NATURAL E APRENDIZADO DE MÁQUINA PARA DECISÕES DE SWING TRADE

Lucas Gama Canto

Projeto de Graduação apresentado ao Curso de Engenharia de Controle e Automação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Heraldo Luís Silveira de Almeida

Rio de Janeiro
Março de 2020

ANÁLISE DE NOTÍCIAS DO MERCADO FINANCEIRO UTILIZANDO PROCESSAMENTO DE LINGUAGEM NATURAL E APRENDIZADO DE MÁQUINA PARA DECISÕES DE SWING TRADE

Lucas Gama Canto

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE AUTOMAÇÃO.

Examinado por:

Prof. Heraldo Luis Silveira de Almeida, D.Sc.

Prof. Flavio Luís de Mello, D.Sc.

Prof. Diego Leonel Cadette Dutra, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2020

Gama Canto, Lucas

Análise de Notícias do Mercado Financeiro Utilizando Processamento de Linguagem Natural e Aprendizado de Máquina Para Decisões de Swing Trade/Lucas Gama Canto. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2020.

XIV, 54 p.: il.; 29, 7cm.

Orientador: Heraldo Luís Silveira de Almeida

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Controle e Automação, 2020.

Referências Bibliográficas: p. 44 – 47.

1. Aprendizado de Máquina. 2. Processamento de Linguagem Natural. 3. Mercado Financeiro. I. Silveira de Almeida, Heraldo Luís. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Controle e Automação. III. Título.

Ao povo brasileiro, pela total contribuição em minha graduação.

Agradecimentos

Gostaria de agradecer a todas as pessoas e situações que tornaram este momento possível. Em especial, meus pais Benedita e Manoel, pelo suporte e esforço incondicional em apoiar minha decisão de vir estudar engenharia no Rio de Janeiro, aos professores da graduação, que me fizeram evoluir no âmbito acadêmico, profissional e pessoal, em especial ao meu orientador e professor Heraldo, que não mediu esforços para me ajudar neste trabalho, e aos amigos que me apoiaram e participaram do meu processo de graduação.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Automação.

ANÁLISE DE NOTÍCIAS DO MERCADO FINANCEIRO UTILIZANDO PROCESSAMENTO DE LINGUAGEM NATURAL E APRENDIZADO DE MÁQUINA PARA DECISÕES DE SWING TRADE

Lucas Gama Canto

Março/2020

Orientador: Heraldo Luís Silveira de Almeida

Curso: Engenharia de Controle e Automação

Com o objetivo de automatizar análises fundamentalistas de mercado, o uso de tecnologia para processamento de texto vem sendo utilizado constantemente no meio acadêmico[1] e profissional[2]. De forma a contribuir para este campo em crescimento, este trabalho discorre um estudo acerca da criação de modelos preditivos sobre a valorização ou desvalorização de ações na bolsa de valores do Brasil (B3, antiga Bovespa) a partir de notícias sobre o mercado brasileiro de forma a auxiliar decisões de Swing Trade, ou seja, compra e venda de ações dentro de uma janela de tempo maior que um dia.

Para isto, o presente projeto utiliza o framework PyText, que se baseia em conceitos de Aprendizado de Máquina, Redes Neurais e Processamento de Linguagem Natural de forma a desenvolver modelos preditivos com a tarefa de classificação textual.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

FINANCIAL MARKET NEWS ANALYSIS USING NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING FOR SWING TRADE DECISIONS

Lucas Gama Canto

March/2020

Advisor: Heraldo Luís Silveira de Almeida

Course: Automation and Control Engineering

In order to automate fundamental market analysis, the use of text processing technology has been constantly used in academic[1] and professional[2] means. To contribute to this growing field, this paper discusses a study about the creation of predictive models regarding the valuation or devaluation of shares on the Brazilian stock exchange (B3, former Bovespa) based on news about the Brazilian market in order to assist Swing Trade decisions, that is, buying and selling stocks within a time window longer than one day.

To this end, the present project uses the PyText framework, which is based on Machine Learning, Neural Networks and Natural Language Processing concepts in order to develop predictive models with the task of textual classification.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AF	Análise Fundamentalista
API	*Application Programming Interface*
AT	Análise Técnica
B3	Brasil, Bolsa, Balcão
BOW	*Bag-of-Words*
BRNN	*Bidirectional Recurrent Neural Networks*
CNN	*Convolutional Neural Network*
CSV	*Comma-separated values*
CUDA	*Compute Unified Device Architecture*
GPU	*Graphics Processing Unit*
HME	Hipótese do Mercado Eficiente
iBovespa	Índice Bovespa
JSON	*JavaScript Object Notation*
LSTM	*Long Short-Term Memory*
MLP	*Multilayer Perceptron*
NILC	Núcleo Interinstitucional de Linguística Computacional
NN	*Neural Networks*
PLN	Processamento de Linguagem Natural
ReLU	*Rectified Linear Unit*
RNN	*Recurrent Neural Network*

TSV	*Tab-separated values*
USP	Universidade de São Paulo

Capítulo 1

Introdução

1.1 Tema

O tema deste trabalho se resume no estudo da criação de modelos preditivos de modo que estes possam prever a valorização ou desvalorização de ações da bolsa de valores por meio do processamento de notícias do mercado brasileiro.

Deste modo, o problema a ser abordado é a identificação de quando uma notícia pode impactar positivamente ou negativamente a variação de preço de ações de forma automatizada.

1.2 Delimitação

Este trabalho se restringe ao processamento de texto em português brasileiro, tendo como foco a predição da variação de preço das ações que fazem parte da bolsa de valores do Brasil, a B3. Pela indisponibilidade de dados sobre notícias brasileiras contendo a informação do horário de lançamento da notícia, o projeto mira em predições dentro de uma janela de tempo maior que um dia, de forma a auxiliar decisões de Swing Trade, isto é, operações de compra e venda de ações numa janela de tempo maior que um dia.

Além disso, o estudo se baseia na ferramenta PyText, um framework recentemente desenvolvido pelo Facebook que providencia modelos de processamento de linguagem natural de última geração através de uma interface simples e extensível[16]. O uso da ferramenta foi motivado principalmente pelo trabalho feito por ALVES, V. A. em [17], onde o PyText é utilizado numa tarefa de classificação de texto para detectar o gênero de letras musicais em português brasileiro.

1.3 Justificativa

Diante do crescente número de investidores na bolsa de valores no Brasil, nota-se uma maior preocupação da população brasileira acerca da busca por independência financeira e fontes alternativas de renda com o intuito de contribuir à economia familiar, previdência, ou mesmo utilizar este método como fonte principal de renda[18].

Ao mesmo tempo, estudos associados à inteligência artificial, aprendizado de máquina e processamento de linguagem natural continuam emergindo no meio acadêmico e auxiliando o meio profissional como nunca antes, incluindo o mercado financeiro[19].

Através destes dois fatores, o presente trabalho busca contribuir para a difusão do estudo e uso de algumas destas tecnologias sobre um assunto que gradualmente se encontra dentro do interesse da população brasileira e que colabora para uma possível instauração de uma cultura de economia e independência financeira no Brasil.

1.4 Objetivos

O objetivo geral do presente trabalho é de analisar modelos preditivos associados ao mercado financeiro que possam ser construídos a partir do framework PyText, tendo como objetivos específicos, apresentar: (1) A busca por dados de notícias e do histórico da bolsa de valores; (2) A lógica utilizada para a união destes dados de forma a construir os conjuntos de dados utilizados no treinamento dos modelos; (3) O pré-processamento dos conjuntos de dados; (4) As possíveis configurações do framework utilizado de forma a obter uma performance razoável; (5) O detalhamento e a análise dos modelos finais encontrados.

1.5 Metodologia

O trabalho teve início a partir da procura por bases de dados de notícias associadas ao mercado brasileiro e escritas em português do Brasil, seguida pela obtenção do histórico das variações de preço dos ativos que compõem o iBovespa. Após isto, o histórico foi filtrado de forma a manter as informações dos 5 ativos mais significativos e das varições destes ativos que ocorreram dentro da mesma janela de tempo das notícias obtidas. Em seguida, estes dados foram unidos de forma a obter 5 conjuntos de dados para cada ativo, cada um levando em consideração uma diferente janela de tempo para indicar a valorização: de um a cinco dias.

Logo após, houve a etapa de pré-processamento do corpo das notícias de forma a remover possíveis ruídos e facilitar a etapa de treinamento, sem perda de contexto do conteúdo. Com os conjuntos de dados prontos, foram feitos testes no PyText com o objetivo de definir a melhor configuração para a natureza dos dados, e assim obter uma performance razoável.

Por fim, os testes finais de cada modelo gerado foi detalhado e analisado para permitir uma conclusão e avaliação do processo como um todo.

1.6 Descrição

O capítulo 2 apresenta toda a fundamentação teórica utilizada como base para o projeto a partir de uma breve descrição de como a bolsa de valores funciona seguida de explicações sobre Aprendizado de Máquina, Processamento de Linguagem Natural, Redes Neurais e o framework Pytext.

No capítulo 3 é detalhado todo o processo executado para obtenção do conjunto de notícias e do histórico da B3, seguido do pré-processamento realizado nestes dois conjuntos e a criação dos conjuntos de dados finais utilizados para o treino, cada um associado a um ativo e uma janela de tempo específica.

Os detalhes das configurações utilizadas no PyText e o treinamento em si é especificado no capítulo 4, onde há uma discussão acerca dos parâmetros encontrados para a geração de modelos mais performáticos, além das métricas finais encontradas para cada modelo gerado seguido de uma breve análise.

Por fim, o capítulo 5 apresenta uma conclusão acerca dos modelos encontrados e do trabalho como um todo, apresentando também sugestões que futuramente podem ser aplicadas para a evolução do tema e uma possível melhora de desempenho dos modelos preditivos.

O código desenvolvido para a geração e pré-processamento dos conjuntos de dados e para geração e execução dos arquivos de configuração do PyText utilizados para a geração dos modelos podem ser encontrados no repositório do github referenciado em [20].

Capítulo 2

Fundamentação Teórica

2.1 Bolsa de Valores e Ações

A Bolsa de Valores é um lugar centralizado onde, além de abranger outros tipos de investimento, se negociam ações (também chamados de ativos ou papéis), isto é, parcelas do capital social de empresas de capital aberto. Atualmente a B3 (Brasil, Bolsa, Balcão) é a Bolsa de Valores oficial do Brasil que em 2017 atingiu a 5ª posição das maiores bolsas de mercados de capitais do mundo em valor de mercado, com um patrimônio de US$ 13 bilhões[21].

As ações são negociadas diariamente a partir das ordens de compra e venda emitidas pelas corretoras durante o pregão eletrônico, que na B3, funciona em dias úteis das 10:00 às 17:00.

2.1.1 Preços de Ações

O preço de um ativo na Bolsa de Valores pode ser determinado por diversas razões que podem se relacionar entre si, entre essas, pode-se destacar a lei da oferta e demanda, perspectivas de crescimento da empresa associada ao papel e especulação. A previsibilidade acerca de movimentações no mercado de ações normalmente pode ser baseada em Análise Técnica (AT - estudo dos movimentos do mercado baseado em métricas como preço, volume e taxa de juros[22]), Análise Fundamentalista (AF - estudo feito a partir de resultados financeiros e operacionais, indicando a saúde da empresa[23]) ou em uma junção destes dois conceitos.

A validade da previsibilidade destas movimentações são questionadas por críticas com base na Hipótese do Mercado Eficiente (HME) e seus três níveis definidos em [24]:

- HME fraca: Afirma que os preços atuais refletem totalmente a informação contida na sequência histórica dos preços. Ou seja, a AT não consegue prever os movimentos futuros pois os preços passados só podem descrever o presente.
- HME semi-forte: Afirma que os preços presentes não só refletem toda a sequência histórica de preços mas também toda a informação pública sobre as organizações associadas ao ativo em questão. Neste nível de eficiência, a AF também não seria capaz de prever movimentos futuros, pois toda informação pública, como demonstrativos de resultados ou análises orçamentárias de uma empresa, refletiria apenas o preço presente.
- HME forte: Neste nível, é afirmado que *toda* informação conhecida sobre as organizações é totalmente refletida pelo preço presente, logo, nem mesmo aqueles com informações privilegiadas podem utilizar isto como ferramenta para prever preços futuros.

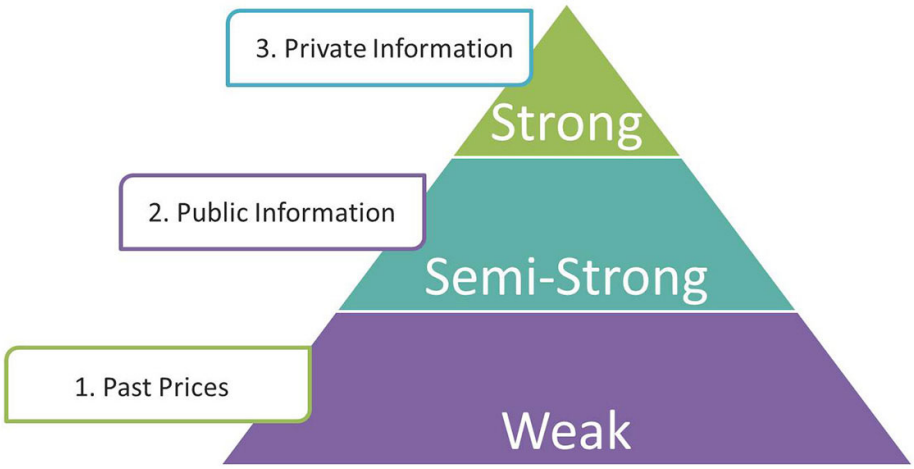


Figura 2.1: Os três níveis da HME, cada nível adiciona um tipo de informação cujo com o qual não seria possível prever um movimento de preço no mercado[3].

Não há uma reposta correta perante a validade da HME. Porém, muitos acadêmicos acreditam, pelo menos, na HME fraca[25], fazendo com que, em algumas ocasiões, seja preferível a utilização da AF, ou seja, a análise de resultados financeiros, relatórios anuais e notícias divulgadas acerca do mercado financeiro.

2.1.2 Índice de Bolsa de Valores

Com objetivo de parametrizar algumas informações intrínsecas às bolsas, estas disponibilizam diversos índices. O principal índice da B3 é o Ibovespa, que é formado pelos ativos com maior volume negociado na bolsa nos últimos meses e indica, de forma resumida, o desempenho das ações negociadas na B3. Por ser um indicador principal, muitos fundos de investimento baseados no mercado de ações estão atrelados ao Ibovespa, contribuindo para a atratividade destes ativos de maneira geral.

Tabela 2.1: Os 5 ativos com o maior volume e participação na B3, associados ao Ibovespa[15].

Código	Ação	Qtde. Teórica	Part.(%)
ITUB4	Itaú Unibanco	4.738.562.684	9,095
PETR4	Petrobras	4.520.185.835	7,038
BBDC4	Bradesco	3.873.597.664	7,028
VALE3	Vale S.A.	3.147.743.563	8,414
ABEV3	AMBEV	4.344.066.764	4,173

2.2 Aprendizado de Máquina

Pode-se definir Aprendizado de Máquina como o campo de estudo de algoritmos com o objetivo de fazer com que computadores possam agir sem serem explicitamente programados para fazer determinada tarefa. São algoritmos que analisam dados e aprendem com eles, gerando um modelo preditivo que pode fornecer uma predição de algo no mundo.

Outra definição dada por Tom M. Mitchell[26] fala que o campo de Aprendizado de Máquina busca responder a pergunta: "Como podemos construir sistemas de computadores que possam automaticamente melhorar através de experiência e quais são as leis fundamentais que governam todo este processo de aprendizado?". Outra definição do mesmo autor[27] diz que "Um programa de computador é dito aprender com a experiência E em respeito a uma tarefa T e medida pelo desempenho P se o seu desempenho em T, medido por P, melhora com a experiência E". Neste conceito, se fosse desejado um programa de computador que aprendesse a classificar e-mails como spam ou não, por exemplo, poderíamos fazer a seguinte associação:

- E = A experiência de ver o usuário classificar emails como spam ou não.
- T = A tarefa de classificar os emails.
- P = O número ou fração de emails corretamente classificados como spam/não spam

Geralmente, os algoritmos de Aprendizado de Máquina podem ser divididos em dois tipos: Aprendizado Supervisionado e Aprendizado Não Supervisionado.

2.2.1 Aprendizado Supervisionado

Neste tipo, o algoritmo é inicialmente servido por uma série de dados rotulados cujo resultado já é conhecido. A ideia é que o algoritmo aprenda a criar uma estratégia para chegar ao resultado baseando-se nesses dados de modelo inicial. O aprendizado supervisionado pode ser dividido em problemas de regressão ou classificação.

Problema de Regressão

Neste tipo de problema, os dados de entrada (parâmetros) são mapeados em uma função contínua. Por exemplo, um algoritmo cujo objetivo fosse prever o preço dos imóveis na cidade do Rio de Janeiro baseando-se em dados como área útil, bairro, número de vagas na garagem, etc é considerado como um problema de regressão, pois o resultado final será um número contínuo, neste caso, o preço dos imóveis.

Problema de Classificação

Neste caso, os parâmetros são mapeados de forma a classificar os dados em categorias distintas. Por exemplo, um algoritmo utilizado para prever se um tumor é benigno ou maligno a partir de dados como o tamanho, rugosidade do tumor e idade do paciente é considerado um problema de classificação, pois o resultado final será a categoria na qual o tumor pertence.

Os problemas de classificação muitas vezes apresentam desbalanceamento de classes no conjunto de dados utilizado para o treinamento do modelo, ou seja, o conjunto de dados pode apresentar poucas amostras de uma determinada classe em relação às outras envolvidas, o que pode ocasionar falhas na predição da classe minoritária.

De forma a corrigir tal problema, existem algumas técnicas que podem ser baseadas em dois conceitos: *Undersampling* e *Oversampling*. A primeira se resume na remoção de amostras das classes majoritárias, e a segunda, no acréscimo de amostras da classe minoritária a partir das amostras já existentes no conjunto, de forma a se obter um conjunto de dados balanceado.

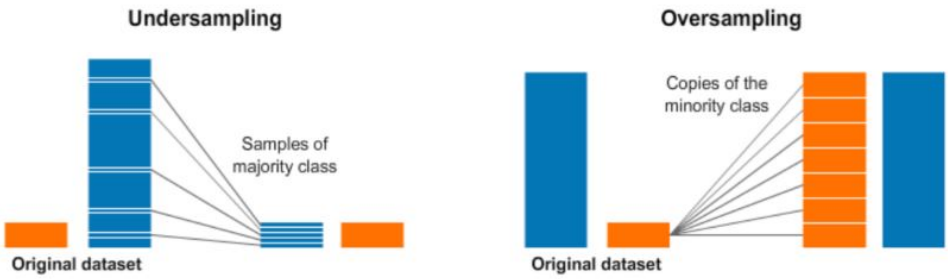


Figura 2.2: Exemplos das estratégias de *Undersampling* e *Oversampling* em um problema de classificação de 2 classes desbalanceadas[4].

Entretanto, estas técnicas podem gerar alguns efeitos negativos, como por exemplo, a demasiada diminuição do conjunto de dados como um todo no *Undersampling*, o que prejudica o aprendizado. No caso do *Oversampling*, pode-se gerar um maior efeito de sobre-ajuste no modelo preditivo, isto é, pela falta de generalização, o modelo consegue prever muito bem amostras do conjunto de dados de treino, mas se mostra ineficaz ao prever dados de teste.

2.2.2 Aprendizado Não Supervisionado

Neste tipo de aprendizado, não existe um conjunto inicial de dados e resultados, ou seja, nos permite abordar problemas onde temos pouca ou nenhuma ideia do que nossos resultados devem aparentar. Um exemplo, seria um algoritmo onde, utilizando uma coleção de 1000 artigos publicados por uma universidade, fizesse um agrupamento de temas desses artigos, baseando-se em diferente variáveis como frequência de palavras semelhantes, número de paginas, etc.

2.2.3 Avaliação de Desempenho

A avaliação de desempenho de um modelo preditivo pode ser realizada através de diversas métricas que são medidas diante de uma previsão do modelo sobre um conjunto de dados de teste, logo, existe uma necessidade em dividir o conjunto de dados inicial em dados de treino e dados de teste.

Não existe um modo ideal de dividir o conjunto de dados, o tamanho do conjunto de treino normalmente é maior que o de teste, de modo que este consiga abranger mais generalizações acerca dos parâmetros do modelo. Assim, algumas proporções são mais comumente usadas, como 60/40, 75/25 e 80/20, proporção baseada no Principio de Pareto que afirma que, 80% das saídas/consequências vem de 20% das

entradas/causas[28]. Neste caso, levando em consideração o escopo de Aprendizado de Máquina, podemos dizer que 20% pode mapear 80% do conjunto de dados.

Além disso, em modelos mais simples, também existe a possibilidade de utilizar a Validação Cruzada, uma técnica que separa o conjunto de dados em subconjuntos exclusivos e diferentes e alguns destes subconjuntos são utilizados para treino e outros para teste, de forma iterativa. Um dos métodos de validação cruzada mais famosos é o *k-fold*, onde o conjunto de dados é separado em k subconjuntos e o treino é realizado k vezes, cada vez utilizando um subconjunto diferente para teste e o resto para treino[29]. No final, as métricas de avaliação são definidas como a média diante dos k subconjuntos.

Métricas

As métricas utilizadas para avaliação dependem do tipo de problema. Por exemplo, em problemas de regressão é comum utilizar o erro quadrático médio[30]. Nos problemas de classificação, é comum utilizar as seguintes medidas: Acurácia, Precisão, Cobertura e a Medida F1:

- Acurácia: É a medida que define a assertividade do modelo em geral, se resume na porcentagem de acertos dentre todas as previsões feitas no conjunto de teste.

$$A = \frac{\text{Número total de acertos}}{\text{Número total de palpites}}$$

- Precisão: Medida de assertividade referente a uma classe específica. É a porcentagem de acertos dentre todos os palpites de uma classe.

$$P_X = \frac{\text{Número total de acertos da classe X}}{\text{Número total de palpites da classe X}}$$

- Cobertura: Porcentagem de palpites certos dentro do número de amostras de uma classe específica.

$$C_X = \frac{\text{Número total de acertos da classe X}}{\text{Número total de amostras da classe X}}$$

- Medida F1: Média harmônica entre Precisão e Cobertura.

$$F1_X = 2\frac{P_X C_X}{P_X + C_X}$$

2.3 Redes Neurais

Redes neurais (*Neural Networks* - NN) são estruturas matemáticas baseadas no funcionamento do cérebro humano. O campo que estuda a aplicação de redes neurais com várias camadas de processamento em métodos de Aprendizado de Máquina é chamado de Aprendizagem Profunda (do inglês, *Deep Learning*). Em sua forma mais simples, uma rede neural contém três camadas: entrada (*input layer*), camada oculta (*hidden layer*) e saída (*output layer*), onde a camada oculta pode ser única ou múltipla. Cada camada é composta por neurônios, também chamados de nós.

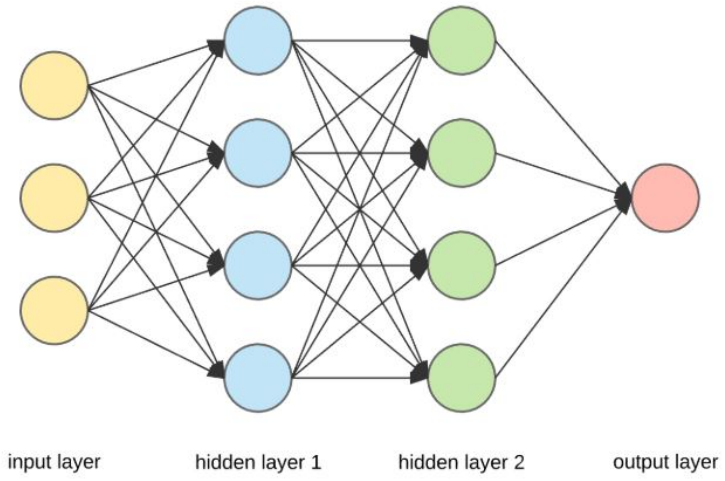


Figura 2.3: Exemplo de rede neural com duas camadas ocultas[5].

Cada nó de uma rede neural representa uma abstração matemática, esta é definida pela função de ativação (*Activation Function*) que recebe o somatório das entradas (*Inputs*) multiplicadas por seus respectivos pesos e mais um viés (*Bias*).

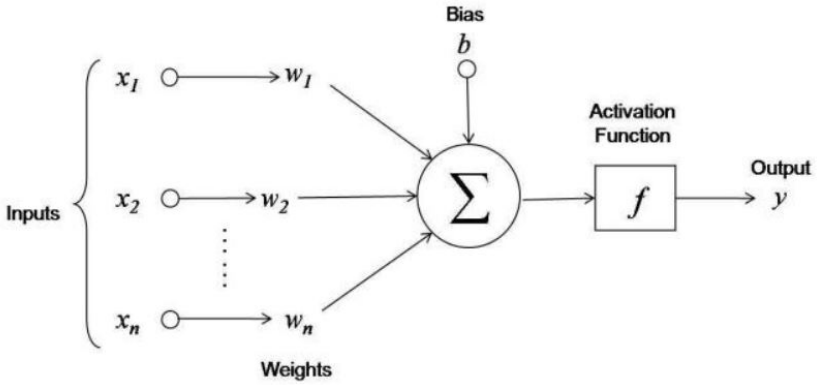


Figura 2.4: Neurônio de uma rede neural[6].

Durante a etapa de treinamento, estes pesos são ajustados de forma que se obtenha saídas iguais ou suficientemente próximas às saídas do conjunto de dados utilizado para treino, isso pode acontecer através de diversas técnicas, sendo a *backpropagation* umas das mais conhecidas, que utiliza descida de gradiente[31] para minimizar a função custo determinada pelas entradas, viés, função de ativação e saída dos neurônios.

Este modelo simples de rede neural com apenas uma camada oculta se chama *Perceptron* e os modelos com múltiplas camadas são chamados de *Multilayer Perceptron* (MLP). Apesar de terem significante poder preditivo, estas redes podem apresentar falhas, principalmente quando o número de camadas ocultas alcança um valor muito grande ou quando os dados de entrada apresentam uma alta dimensão. Mesmo após algumas tentativas de solucionar estas falhas das redes MLP, outras arquiteturas de redes neurais foram propostas.

2.3.1 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (*Convolutional Neural Network - CNN*) foram desenvolvidas com o intuito de facilitar a classificação de dados de alta dimensão, como imagens e textos. Em um problema de classificação de imagem com uma rede MLP, cada pixel está diretamente ligado a uma entrada, ocasionando um vetor de entrada de alta dimensão. Em uma rede CNN, normalmente, este vetor de entrada sofre diversas reduções de dimensionabilidade dentro de três camadas: camada de convolução, camada de função de ativação e camada de *pooling*.

Na primeira camada, ocorre a convolução do dado de entrada com um filtro (*kernel*), que também é treinado ao longo do treinamento de toda a rede.

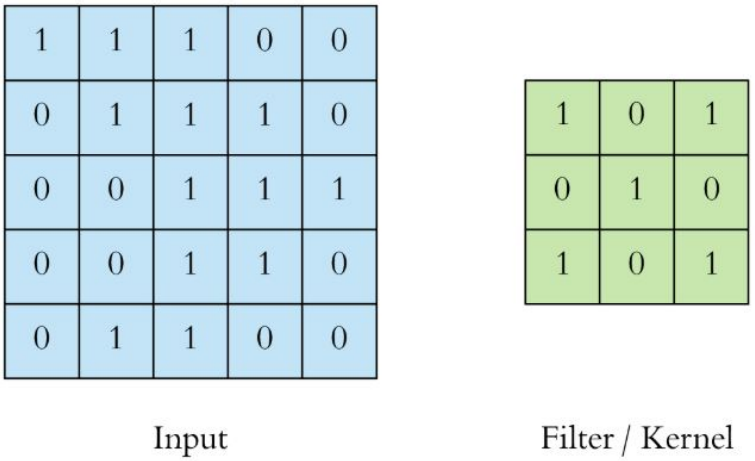


Figura 2.5: Exemplo de um dado de entrada e um *kernel*[7].

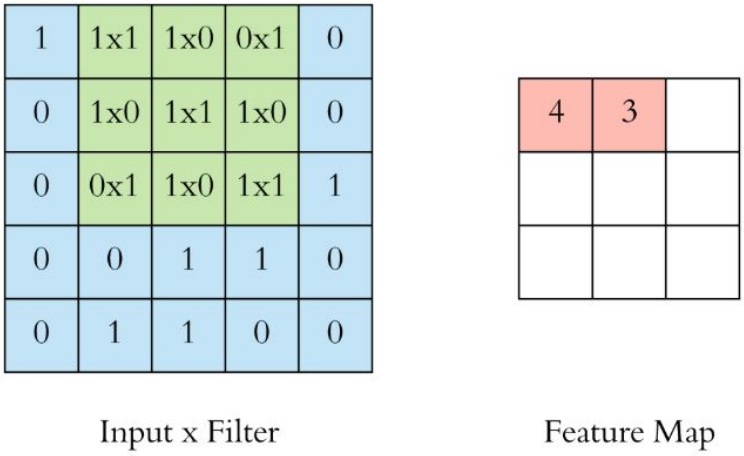


Figura 2.6: Exemplo de convolução da imagem anterior[7].

Após esta etapa, o resultado da camada convolução passa por funções de ativação. Nestas, normalmente são atribuídas funções ReLU (*Rectified Linear Unit*), análoga à função rampa, ou seja, valores menores que zero são zerados e o resto é mantido. Por final, o resultado da camada de função de ativação passa pelo *pooling* onde a redução de dimensibilidade pode ser feita através de diversas técnicas, sendo a *max-pooling* a mais utilizada. Esta consiste em selecionar os maiores valores dentro de submatrizes de tamanho e espaçamento (passos) específicos, como indicado na figura 2.7.

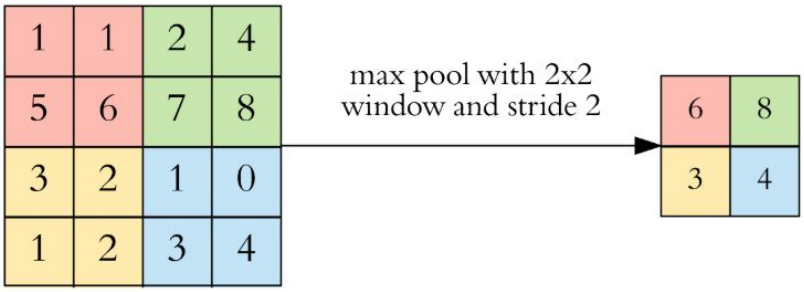


Figura 2.7: Exemplo de *max-pooling* utilizando uma submatriz de tamanho 2x2 e passo 2[7].

Após estas camadas, o resultado pode ser utilizado na camada de entrada de uma rede comum, normalmente, uma rede MLP.

2.3.2 Redes Neurais Recorrentes

As Redes Neurais Recorrentes (*Recurrent Neural Networks - RNN*) são redes que foram desenvolvidas principalmente para problemas associados à predição de sequên-

cia de dados, como a predição de palavras em um texto de entrada e reconhecimento de fala[32]. A grande diferença destas para redes neurais comuns é que, além das entradas tradicionais (saída de uma camada de entrada ou a saída de uma camada oculta anterior), a saída de um estado anterior de cada neurônio pode ser utilizada como uma entrada, fazendo assim uma realimentação de informação. Desta forma, os dados de entrada são inseridos na rede de forma sequencial.

Como mostrado em [33], uma rede RNN possui vantagens como a consideração da informação histórica durante o processamento e a possibilidade de processar entradas de qualquer tamanho, mas também possui algumas desvantagens como processamento lento e a alta ocorrência do problema de dissipação do gradiente (*Gradient Vanishing*), que ocorre pelo fato de que, em RNNs, é difícil de capturar dependências de longo prazo devido ao gradiente multiplicativo, que pode crescer/decrescer de forma exponencial, de acordo com o número de camadas na rede.

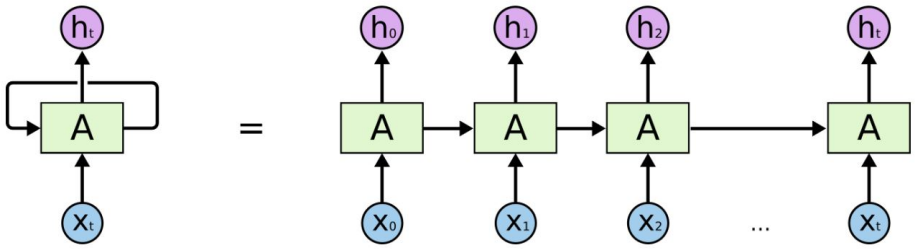


Figura 2.8: A célula de uma RNN (à esquerda) pode ser descrita como uma série de neurônios se comunicando entre si através de passos no tempo (forma "desdobrada")[8]. É ao longo destes passos que o problema de dissipação de gradiente ocorre.

Com o intuito de lidar com o problema de dissipação do gradiente, foi criada a arquitetura LSTM (*Long Short-Term Memory*)[34], que contém células com a capacidade de armazenamento de memória (como pode ser visto na figura 2.9), ocasionando uma melhora no aprendizado em uma rede RNN.

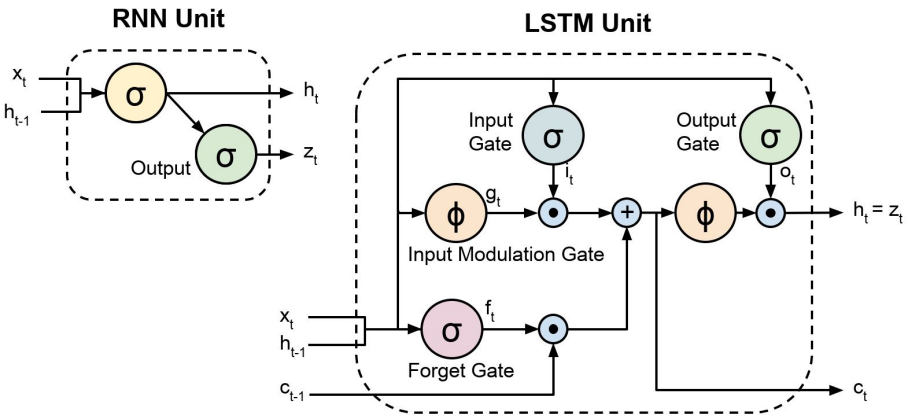


Figura 2.9: Diferença entre a célula de uma rede RNN comum e de uma rede LSTM [9].

Nas RNN comuns e na LSTM, o dado é inserido de forma sequencial em apenas um sentido, fazendo com que a rede obtenha informação sequencial a partir do passado. De forma a obter informação sequencial proveniente do futuro, criou-se o conceito de redes neurais recorrentes bidirecionais (*Bidirectional Recurrent Neural Networks* - BRNN). Neste tipo de rede, o algoritmo é alimentado pelo conjunto de dados de duas formas paralelas: do início ao fim e do fim ao início, combinando estas duas em uma única saída.

Uma das BRNNs mais conhecidas é a versão bidirecional da LSTM (*Bidirectional Long Short-Term Memory* - BiLSTM) que une a funcionalidade de memória da LSTM com o benefício bilateral das BRNNs.

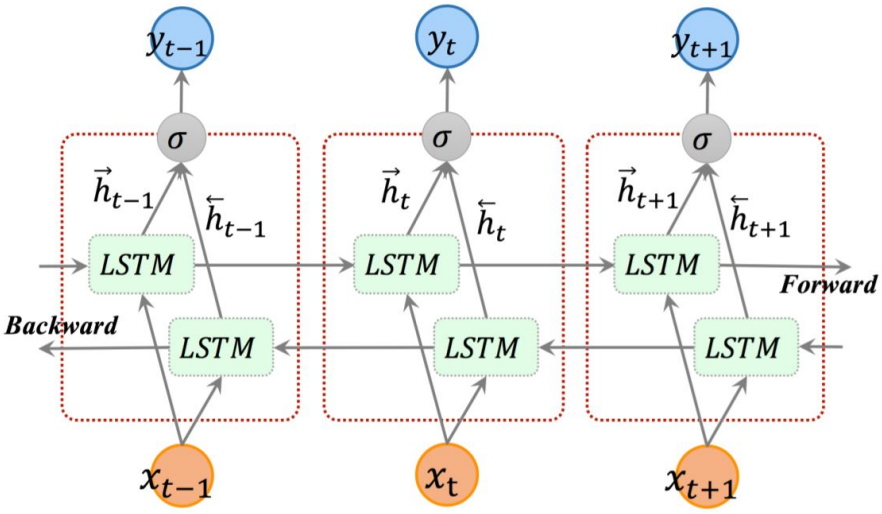
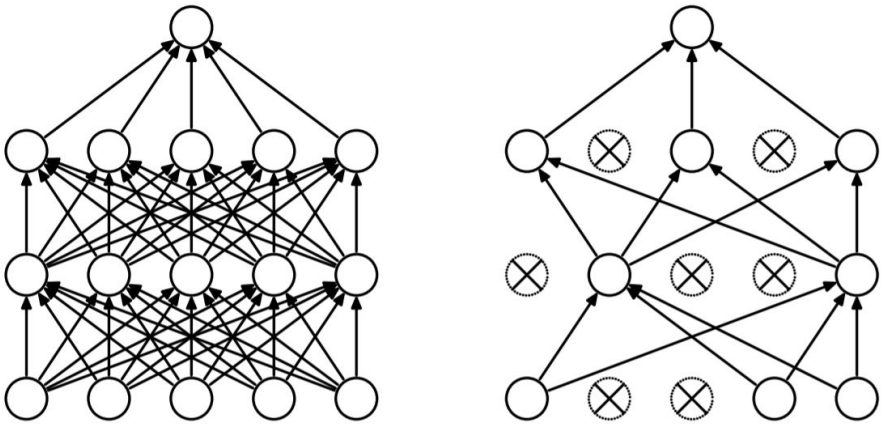


Figura 2.10: Arquitetura de uma rede BiLSTM na forma "desdobrada".[10].

2.3.3 *Dropout*

Dropout é uma técnica para prevenção de sobre-ajuste patenteada pelo Google e apresentada em [11]. Esta técnica se baseia no desligamento temporário e aleatório de alguns neurônios da rede ao longo da etapa de treinamento.



(a) Standard Neural Net (b) After applying dropout.

Figura 2.11: Comparação entre uma NN comum e a mesma após a aplicação de *dropout*.[11].

O efeito do *dropout* pode ser definido por um parâmetro p que indica a probabilidade de um neurônio de permanecer na rede.

2.4 Processamento de Linguagem Natural

Processamento de Linguagem Natural (PLN) se resume ao campo de estudo das tecnologias utilizadas para ajudar computadores a entenderem a linguagem natural dos humanos, é também considerado uma subárea da Inteligência Artificial. Pode ser usado em diversas aplicações[35], como por exemplo:

- Aplicativos de tradução de idioma, como o Google Translator
- Processamento de palavras, que empregam PLN para a correção gramática
- Resposta de voz iterativa em *call centers*, de forma a responder adequadamente conforme a requisição de usuários
- Assistentes pessoais, como OK Google, Siri, Cortana e Alexa

Geralmente, o PLN abrange um pré-processamento de texto antes deste ser transformado em uma forma inteligível por computadores, de forma a remover ruídos ou facilitar o processamento, e isto pode ocorrer de diversas formas.

2.4.1 Pré-processamento de Sintaxe

A sintaxe se refere à forma de como as palavras se organizam em uma sentença para que se obtenha sentido gramatical. Estas são algumas técnicas de sintaxe que podem ser utilizadas no pré-processamento de texto:

- Stemização (*Stemming*): É a transformação de palavras flexionadas para sua forma radical. Por exemplo, as palavras "estudos", "estudar" e "estudando" se transformariam apenas em "estud", mas a palavra "tiver"se transformaria em "tiv" e "tenho" se transformaria em "tenh".

- Lematização (*Lemmatization*): Semelhante à Stemização, porém, a palavra é resumida para seu lema, fazendo com que se alcance um nível maior de abstração. Neste caso, tanto a palavra "tiver" como "tenho" se transformaria no lema "ter".

- Remoção de *stopwords*: A remoção de "palavras de parada", ou seja, palavras como "a", "de", "o", "da", "que", "e", "do". É útil pois, na maioria das vezes, não são informações importantes para construção do modelo.

Além destas, outra técnicas mais simples são utilizadas, como a transformação de caracteres maiúsculos para minúsculos.

2.4.2 Representação Vetorial

Após o pré-processamento textual, diversas técnicas podem ser utilizadas para a transformação do texto em números. Uma das formas mais simples de se fazer isso é o *Bag-of-Words* (BOW), que consiste em simbolizar textos de um conjunto de dados através de uma matriz na qual cada coluna é associada a uma palavra existente no conjunto de dados. Cada linha da matriz representa um texto e as variáveis indicam o número de ocorrências de uma palavra específica.

Figura 2.12: Exemplo de aplicação do *Bag-of-Words*[12].

Apesar da praticidade e simplicidade, o BOW apresenta alguns problemas, entre eles, a perda da informação de ordem das palavras e a alta dimensão da representação vetorial.

2.4.3 Word2Vec

o Word2Vec é um conjunto de técnicas utilizadas para a transformação de texto, normalmente utilizando redes neurais, em representações vetoriais chamadas de *embeddings* onde é possível identificar a proximidade entre palavras de forma semântica.

Apresentado inicialmente em [13], o Word2Vec foi introduzido a partir de duas diferentes arquiteturas: CBOW (*Continuous Bag-of-Words*) e Skip-gram. O CBOW consiste na predição de uma palavra que possa estar localizada no meio de um conjunto de palavras (entre palavras do passado e do futuro), enquanto que o Skip-gram tenta prever um conjunto de palavras que podem estar localizadas ao redor de uma palavra específica (palavra do presente).

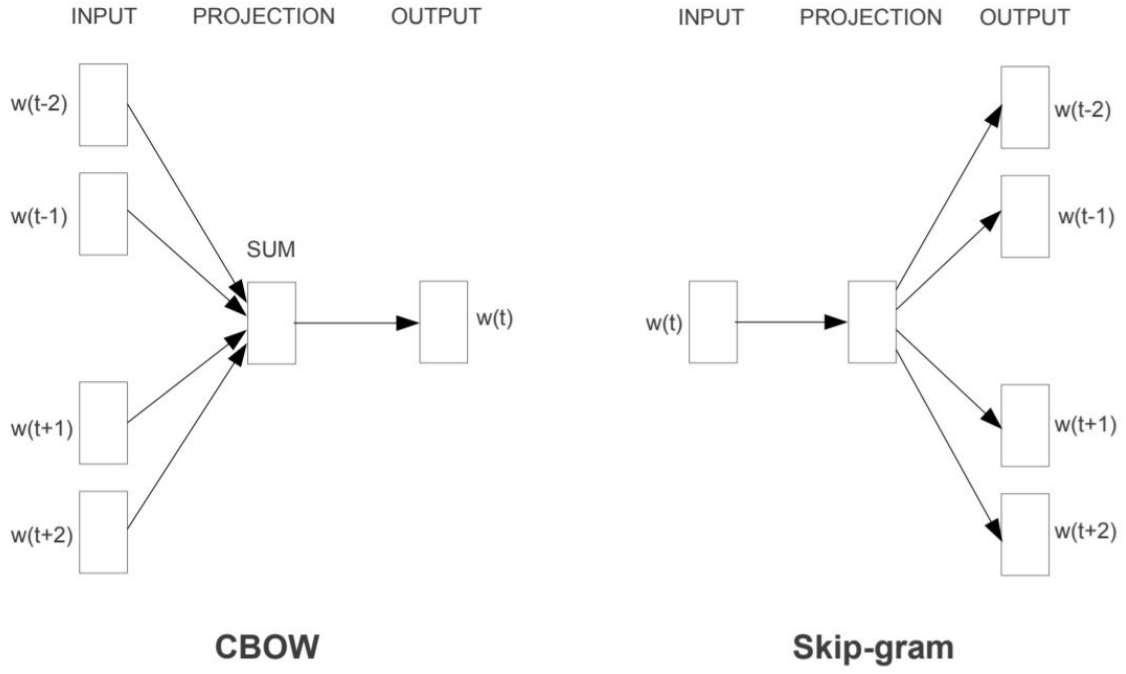


Figura 2.13: Comparação entre a arquitetura CBOW e a arquitetura Skip-gram[13]. O *embedding* é gerado a partir do resultado de treino de uma rede.

As arquiteturas CBOW e Skip-gram serviram de base para a criação de outros modelos do Word2Vec, fazendo com que surgissem repositórios onlines de *embeddings* gerados a partir de diferentes modelos, como o repositório de *embeddings* do NILC (Núcleo Interinstitucional de Linguística Computacional)[36].

2.5 PyText

Lançado no final de 2018 por uma equipe do Facebook, o PyText é um framework para modelagem de PLN baseado em *Deep Learning* e PyTorch, uma biblioteca para aprendizado de máquina. Ele se destaca pelo baixo nível de abstração, pela possibilidade de alterar e servir um modelo preditivo no formato caffe2 em produção e pelo uso de técnicas recentes de *Deep Learning* e PLN como as arquiteturas BiLSTM e CNN para classificação de texto e o uso de *embeddings* para representação vetorial do conjunto de dados textual.

O PyText também possui uma funcionalidade que possibilita o uso do processamento de uma GPU (*Graphics Processing Unit*), caso a máquina onde o PyText esteja sendo executado possua uma, para agilizar o treinamento de modelos através da API CUDA (*Compute Unified Device Architecture*) que está presente em placas de vídeo recentes e habilita o processamento paralelo em GPUs.

Além disso, a ferramenta possui integração com o TensorBoard, uma interface gráfica em forma de aplicação web providenciada pela biblioteca TensorFlow que disponibiliza métricas e gráficos de interesse associados aos modelos em desenvolvimento, abrangendo a etapa de treino e de teste. As métricas são apresentadas durante a fase de treino ao longo de *epochs*, etapas onde os pesos da rede são atualizados após a leitura de todo o conjunto de dados.

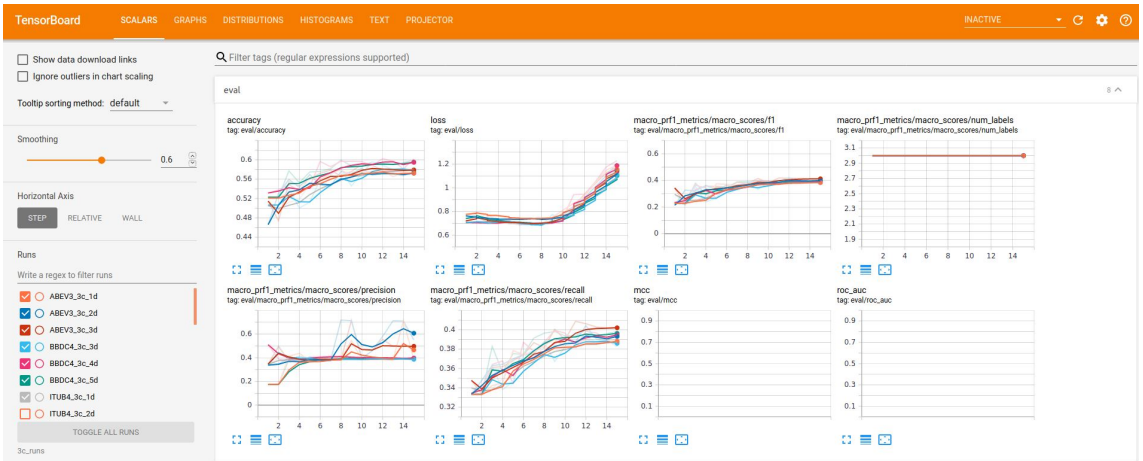


Figura 2.14: Painel do TensorBoard associado a diversos modelos gerados pelo PyText. A interface também apresenta gráficos da arquitetura dos modelos gerados e as projeções dos *embeddings* utilizados.

As especificações do PyText podem ser adquiridas com maiores detalhes na documentação oficial da ferramenta, referenciada em [37]. Uma clara explicação sobre

sua arquitetura, além de uma comparação com diferentes frameworks para PLN pode ser lida em [17].

2.5.1 Instalação em Máquina Virtual Ubuntu

Nesta subseção, serão descritos os comandos utilizados para instalação do PyText, sem a utilização de CUDA, numa máquina virtual Linux, criada pelo software Oracle VM VirtualBox Manager a partir da imagem Ubuntu 18.04.3 LTS, sem a adição de qualquer outro software além das atualizações do sistema operacional.

Inicialmente, foram feitas algumas tentativas de instalação a partir dos comandos demonstrados em [17], mas com isto a ferramenta não demonstrou o funcionamento correto, possivelmente por conta de divergências entre a instalação da ferramenta em uma máquina virtual e em uma máquina física ou entre as versões mais recentes das bibliotecas associadas ao PyText.

Sendo assim, o comandos a seguir foram executados:

```
sudo apt-get update #atualização de pacotes do Linux
sudo apt-get install protobuf-compiler libprotoc-dev #comando recomendado
↪ pela documentação, para distros Ubuntu/Debian[36]
sudo apt install python3-pip #instalação do Python 3
```

Após estes comandos, o comando para a instalação da biblioteca PyTorch foi gerado a partir da página oficial do PyTorch[38] no campo *Run This Command* da seção *Quick Start Locally* ao selecionar as opções: PyTorch Build: Stable, OS: Linux, Package: Pip, Language: Python 3.6 e CUDA: None.

```
pip3 install torch==1.3.1+cpu torchvision==0.4.2+cpu -f
↪ https://download.pytorch.org/whl/torch_stable.html #comando de
↪ instalação do PyTorch gerado no momento do trabalho
pip3 install pytext-nlp #instalação do PyText
pip3 install tensorflow tensorboard #instalação do TensorBoard
```

Após estes comandos, a máquina virtual foi reiniciada e o framework estava pronto para uso.

2.5.2 Configuração e Execução

O PyText é considerado uma ferramenta de interface simples e possui uma alta capacidade de rápida prototipação[16]. Isto se dá principalmente pelo fato de que a configuração de treino, teste e avaliação de um modelo é feita através de um

único arquivo JSON (*Javascript Object Notation*). Neste, é informado o caminho para o tipo de tarefa (classificação de texto ou identificação de entidades) conjunto de dados de teste, treino e avaliação, arquitetura de rede neural que será utilizada para treinamento, caminho para arquivo de *embedding* pré-treinado (se disponível) e outros diversos parâmetros que podem ser ajustados para o treinamento de um modelo específico.

```
{
  "version": 8,
  "task": {
    "DocumentClassificationTask": {
      "data": {
        "source": {
          "TSVDataSource": {
            "field_names": ["label", "slots", "text"],
            "train_filename": "tests/data/train_data_tiny.tsv",
            "test_filename": "tests/data/test_data_tiny.tsv",
            "eval_filename": "tests/data/test_data_tiny.tsv"
          }
        }
      },
      "model": {
        "DocModel": {
          "representation": {
            "DocNNRepresentation": {}
          }
        }
      }
    }
  },
  "export_torchscript_path": "/tmp/new_docnn.pt1",
  "export_caffe2_path": "/tmp/model.caffe2.predictor"
}
```

Figura 2.15: Exemplo de arquivo de configuração do PyText (*docnn.json*), retirado do repositório do GitHub do projeto[14].

Após a configuração, basta rodar um comando do PyText referenciando o arquivo JSON criado para que o treino se inicie:

```
pytext train < demo/configs/docnn.json
```

Após o fim do treino, o arquivo caffe2 gerado é incluido na pasta *caffe2_ exports*, caso a opção de *metric reports* seja ativada na configuração, os arquivos de métricas são incluídos na pasta *metric_ reports* e os arquivos de informações de treino/teste associados ao TensorBoard são incluídos na pasta *runs*.

2.5.3 Visualizando Painel TensorBoard

Para visualizar os dados armazenados na pasta *runs*, a aplicação web do TensorBoard deve ser iniciada já associada à pasta através do seguinte comando:

```
tensorboard --logdir=runs
```

Com isso, a aplicação web é servida na porta 6006 do servidor local, bastando utilizar um navegador para acessar o endereço *http://localhost:6006/*.

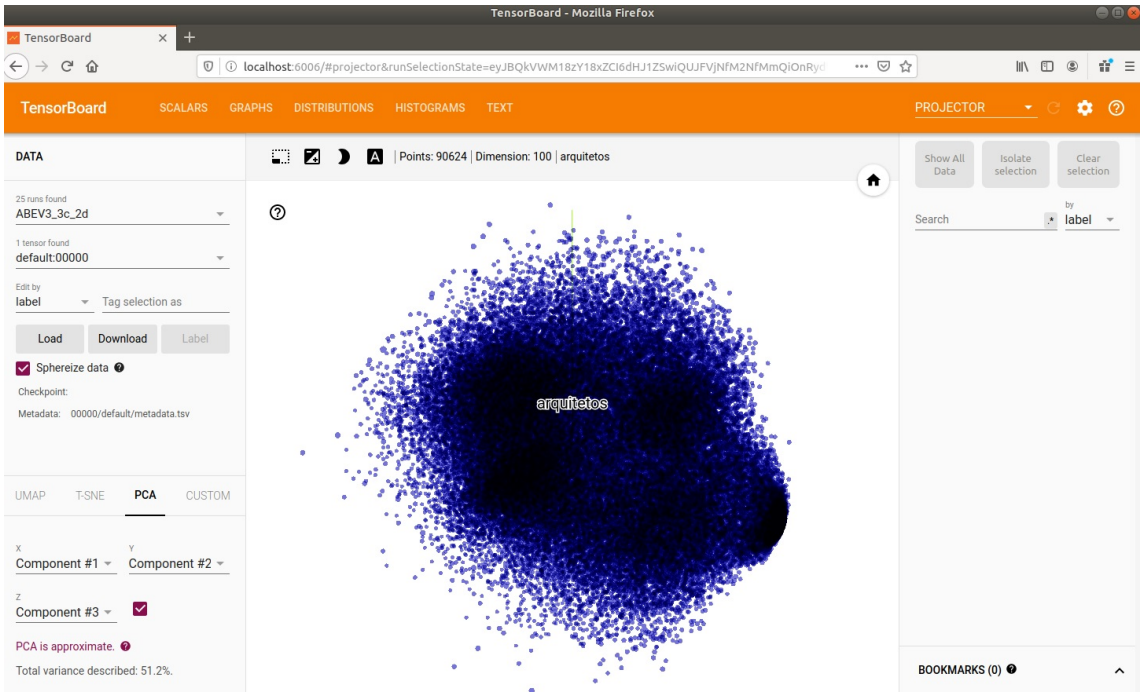


Figura 2.16: Aplicação web do TensorBoard sendo acessada em *http://localhost:6006/* pelo Mozilla Firefox, mostrando a projeção do *embedding* de um modelo.

Capítulo 3

Obtenção e Tratamento de Dados

Baseando-se na HME fraca e acreditando na possibilidade de realizar a análise fundamental de notícias sobre o mercado de forma autônoma, foi decidido o estudo acerca da criação de modelos preditivos de modo a prever a valorização, desvalorização ou preservação do preço de ativos da B3. Para isso, seria necessário a aquisição de um conjunto de dados de notícias e variações de preço de ativos grande o suficiente que relacionasse as datas/horários das notícias com as datas/horários das variações. Como um conjunto de dados que relacionasse diretamente notícias e variações de ativos não foi encontrado, ficou claro que seria necessário o cruzamento de dados a partir de dois conjuntos de dados diferentes.

3.1 Conjunto de Dados de Notícias

Inicialmente, foi preferido a escolha de realizar análises de forma a auxiliar decisões de *Day Trade*, isto é, compra e venda de ações no mesmo dia, o que tornaria a informação de horário imprescindível. Isto logo se tornou inviável devido a falta de conjuntos de dados de notícias em português brasileiro com a informação de horário. Dentre os encontrados, todos continham apenas a data de lançamento da notícia.

Conjuntos de notícias em outros idiomas com a informação de horário foram encontrados, mas não foram utilizados devido ao foco do projeto que buscava analisar notícias do mercado brasileiro para auxiliar operações na B3. Também foram avaliadas as possibilidades de construir um *web crawler* para obter notícias de sites como Folha de S. Paulo[39] ou InfoMoney[40] ou mesmo utilizar a API do Twitter[41] para obter *tweets* de contas associadas a veículos de notícias. A primeira possibilidade se mostrou complexa e demorada para o prazo estipulado para o projeto e a segunda foi impedida pelos limites que a versão gratuita da API propõe sobre os usuários. Sendo assim, foi decidido que a análise seria feita de forma a auxiliar decisões de *Swing Trade*.

Dentre os conjuntos encontrados, foi escolhido o *News of the brazilian newspaper*[42], disponibilizado por um usuário do Kaggle, uma comunidade on-line de cientistas de dados. O conjunto contém 167,053 notícias categorizadas da página web do jornal Folha no período entre janeiro de 2015 e setembro de 2017. O conjunto oferece as informações de, título, corpo, data, categoria, subcategoria e link da notícia.

É importante notar que, além de notícias, o conjunto de dados também continha artigos escritos por colunistas do jornal.

Index	title	text	date	category	subcategory	link
166503	Bancos aumentam praz…	Com a alta dos juros e o men…	2015-04-01	mercado	nan	http://www1.folha.uo…
138160	Professores e alunos da USP…	Funcionários, alunos e prof…	2015-05-14	educacao	nan	http://www1.folha.uo…
112243	Criou-se a ideia de médi…	César Fernandes, 64…	2015-09-27	equilibrioesa…	nan	http://www1.folha.uo…
85392	Manter tumor sob controle …	O tratamento tradicional d…	2016-02-24	equilibrioesa…	nan	http://www1.folha.uo…
28458	Os desafios do agronegócio	"Da porteira para dentro a…	2017-02-19	opiniao	nan	http://www1.folha.uo…
36009	Com choque entre Poderes…	Para evitar um agravamento n…	2016-12-15	poder	nan	http://www1.folha.uo…
93963	Leitores comentam prin…	A Folha dedicou a pri…	2016-04-01	paineldoleitor	nan	http://www1.folha.uo…
48741	Câmera e escuridão aju…	De avião ou de carro, a viag…	2016-09-22	turismo	nan	http://www1.folha.uo…
37322	Trump conversa com Al Gore e…	Donald Trump estava com fr…	2016-06-12	ambiente	nan	http://www1.folha.uo…
115589	Presidente da Colômbia afir…	Semanas após o governo venez…	2015-09-09	mundo	nan	http://www1.folha.uo…
10572	'Tenho condições de …	Em entrevista à rádio Capit…	2017-07-18	poder	nan	http://www1.folha.uo…
99170	Dilma ainda não perdeu	A presidente Dilma Roussef…	2015-03-12	colunas	viniciusmota	http://www1.folha.uo…
77952	'Pronto-socorro é o l…	Diante desse surto inesper…	2016-05-04	colunas	claudiacolluc…	http://www1.folha.uo…
107065	Mourinho é expulso, e Ch…	A má fase do Chelsea parec…	2015-10-24	esporte	nan	http://www1.folha.uo…

Figura 3.1: Alguns exemplos de notícias encontrados no conjunto de dados do jornal Folha de S. Paulo. A figura mostra todas as colunas disponíveis no conjunto: "*title*", "*text*", "*date*", "*category*" e "*subcategory*".

3.1.1 Pré-processamento de Texto

Como o conjunto de dados oferecia a informação de categoria da notícia, estes foram filtrados de forma a adquirir apenas as notícias associadas ao mercado, ou seja, onde os valores da coluna "*category*" fossem iguais a "mercado", o que diminuiu o número total de notícias para 20.970. A coluna "*subcategory*" estava vazia em quase todos os registros, tendo apenas o nome do colunista na maioria dos casos diferentes deste. Os artigos não foram removidos, sendo considerados como notícias no conjunto de dados de forma a contribuir para o volume total.

Alguns testes de treino envolvendo apenas o título da notícia foram feitos, estes demonstraram uma baixa taxa de aprendizagem, fazendo com que se preferisse utilizar o processamento do corpo das notícias.

O Núcleo Interinstitucional de Linguística Computacional (NILC) da USP oferece um repositório[36] contendo uma série de *embeddings* pré-treinados em português brasileiro. Infelizmente, o repositório encontrava-se em manutenção devido a problemas no servidor, o que levou a optar pelo treinamento de *embeddings* nativo do PyText a partir dos dados de treino. Para facilitar e agilizar o treinamento do *embedding* o corpo das notícias foi pré-processado pelas seguintes etapas:

- Normalização para letras minúsculas, para evitar redundâncias.
- Remoção de *stopwords*, palavras como "a", "o", "que" e "de".
- *Stemming*, transformação de palavras flexionadas para sua forma radical.
- Remoção e substituição de caracteres especiais.

Além disso, o formato da coluna de data foi alterada para que esta tivesse a mesma forma da informação de data no conjunto de dados da B3. Por final, as colunas "*title*", "*category*" e "*subcategory*" foram removidas do conjunto.

Index	text	date
116861	terceir maior revendedor caminho pais volv apost tecnolog fre qued vendas mer…	2015-02-09 00:00:00
127905	mercedesbenz volkswagen dev ser primeir empres abc ader program proteca emprego…	2015-07-07 00:00:00
115507	setor priv reag form energ sinaliz ministr joaquim levy fazenda govern pod…	2015-10-09 00:00:00
80976	dol ating menor cotaca nominal sem correca inflacao quas set meses investi…	2016-03-18 00:00:00
126211	falt quorum govern sofr derrot sen tentat aceler aprov med regulariz dinhe…	2015-07-16 00:00:00
109122	propost ab inbev sabmill promet ser alvo investig minuc govern afric sul pass co…	2015-10-14 00:00:00
38647	tesour diret pass segund rod mudanc tentat atra investidor plataform vend t…	2016-11-28 00:00:00
109361	lojist acredit ano marc fim definit black fraude term cunh internaut frustr…	2015-10-13 00:00:00
86106	apos termin mestr engenh ambiental espanha gustav papini 31 sab dificil ac…	2016-02-22 00:00:00
117832	credor galva engenh aprov plan recuper judicial companh assembl realiz nest se…	2015-08-28 00:00:00
147638	contribuint pens burl sistem defes receit dev fic atento pois chanc exit r…	2015-03-26 00:00:00

Figura 3.2: Exemplos de notícias no conjunto de dados após o pré-processamento.

3.2 Conjunto de Dados da B3

Os dados das séries históricas dos ativos pertencentes à B3 pode ser obtido no próprio site da bolsa[43]. Por conveniência, novamente foi decidido usar um conjunto de dados do Kaggle[44], pois este já agrega os dados oriundos de [43] entre janeiro de 2000 e maio de 2018, no formato CSV.

Index	TypeReg	Date	BDICode	Codneg	MarketType	Company	Spec	Prazot	Currency	Open	Max	Min	Med	Close	Preofc	Preofv	Totneg	Quatot
772492	1	20050330	96	CTNM4F	20	COTEMINAS	PN *		R$	210.06	210.06	210.06	21006	210.06	213	260	1	2000
3116286	1	20130905	78	BBASK25	70	BBAS	ON NM	000	R$	0.88	0.93	0.88	90	0.93	0.16	0	2	2700
2554095	1	20120206	78	OGXPC19	70	OGXP	ON NM	000	R$	0.25	0.32	0.25	28	0.3	0.29	0.3	422	2936000
1873496	1	20091013	2	ELET3	10	ELETROBRAS	ON N1		R$	28.1	28.23	27.76	2803	28.23	28.02	28.23	1637	639900
2926036	1	20130314	74	IBOVF67	70	IBOV	IBO	000	R$	114	122	85	11113	95	0	0	8	220
2322434	1	20110503	2	GGBR3	10	GERDAU	ON N1		R$	15.13	15.14	14.8	1493	14.94	14.94	14.98	977	937000
4987374	1	20171221	12	JSRE11	10	FII JS REAL	CI		R$	99.69	100	99.3	9979	100	99.64	100	147	12344
3164345	1	20131021	82	CIELX17	80	CIELE	ON NM	000	R$	1.42	1.42	1.42	142	1.42	0	0	1	150000
3023508	1	20130613	78	PETRG17	70	PETR	PN	000	R$	2.03	2.83	1.99	240	2.46	2.45	2.8	379	698100
4764710	1	20170717	96	BMYB34F	20	BRISTOLMYERS	DRN		R$	174.31	174.31	174.31	17431	174.31	0	0	1	68
3906982	1	20150911	2	CMIG3	10	CEMIG	ON N1		R$	7.42	7.45	7.24	729	7.3	7.3	7.31	238	87700
1357874	1	20071123	96	BRKM5F	20	BRASKEM	PNA N1		R$	14.75	14.75	14.43	1458	14.6	14.48	14.6	24	570
1919458	1	20091207	78	ITUBA42	70	ITUB /ED	PN N1	000	R$	0.86	0.86	0.86	86	0.86	0	0	2	150000
3632307	1	20150108	78	CIELA38	70	CIEL FM	ON NM	000	R$	1.54	1.54	1.54	154	1.54	0	0	3	4900
3202195	1	20131127	8	CCHI3	10	CHIARELLI	ON		R$	0.11	0.14	0.11	12	0.13	0.12	0.13	22	350000

Figura 3.3: Conjunto de dados da B3 antes do tratamento.

O primeiro passo dado para tratar os dados foi a remoção de registros que estão fora da mesma janela de tempo que o conjunto de dados das notícias, seguido da seleção dos 5 maiores ativos presentes no iBovespa, como informado na tabela 2.1. Além disso, os nomes das colunas foram normalizadas para letras minúsculas e foram preservadas apenas as colunas de código do ativo (*codneg*), preço de fechamento (*close*) e data (*date*).

Index	date	codneg	close
4913538	2017-10-31 00:00:00	PETR4	16.77
4900232	2017-10-23 00:00:00	ABEV3	21.3
4435017	2016-11-07 00:00:00	ITUB4	38.07
4918445	2017-11-06 00:00:00	ABEV3	20.54
3880182	2015-08-19 00:00:00	ITUB4	26.28
4318338	2016-08-04 00:00:00	VALE3	18.76
4974905	2017-12-14 00:00:00	ABEV3	20.97
3636853	2015-01-13 00:00:00	BBDC4	34.95
4056369	2016-01-15 00:00:00	BBDC4	17.25
4725435	2017-06-16 00:00:00	ABEV3	17.78
4118516	2016-03-07 00:00:00	BBDC4	26.16

Figura 3.4: Conjunto de dados da B3 após tratamento.

3.3 Cruzamento e Conjuntos Finais

Os conjuntos de dados foram cruzados de forma a obter um conjunto de dados específico para cada combinação de ativo/período de dias. Foi considerado a janela de tempo mínima de 1 dia e uma máxima de 5 dias, logo, teríamos $5 \times 5 = 25$ conjuntos de dados finais (quantidade de ativos $\times$ número de períodos diferentes).

Sendo assim, um conjunto de dados final relacionaria uma notícia à uma classe que indicaria a valorização de um determinado ativo em uma janela de tempo específica. A classificação foi definida da seguinte forma:

- "1": Ocorreu valorização, preço de fechamento futuro é maior que o preço de fechamento passado.
- "0": O valor se manteve, preço de fechamento futuro é igual ao preço de fechamento passado.
- "-1": Ocorreu desvalorização, preço de fechamento futuro é menor que o preço de fechamento passado.

O preço de fechamento passado foi definido como o preço de fechamento do ativo no dia útil anterior ao dia do lançamento da notícia, ao passo que o preço de fechamento futuro foi definido como o preço de fechamento do ativo X dias úteis depois do lançamento da notícia, sendo X um número inteiro entre 1 a 5.

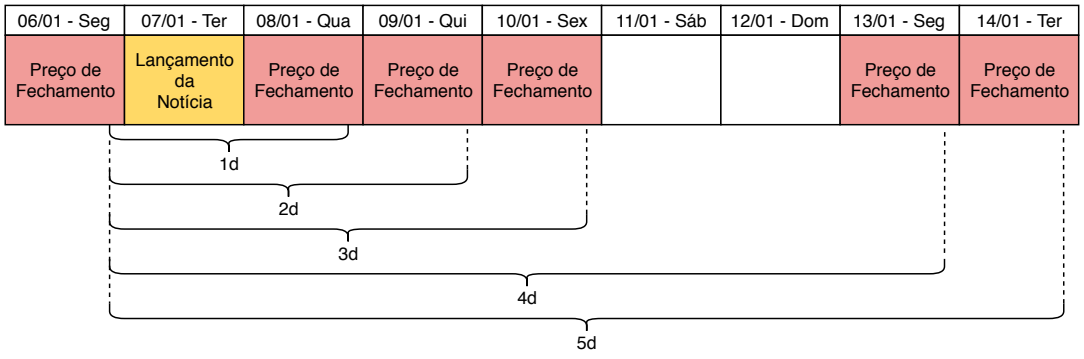


Figura 3.5: Esquema de comparação entre preços de fechamento para a definição da classe de valorização para os intervalos de $1d$ (um dia), $2d$ (dois dias), $3d$ (três dias), $4d$ (quatro dias) e $5d$ (cinco dias).

Após a criação dos conjuntos de dados para cada dupla de ativo/período, o balanceamento entre classes foi analisado.

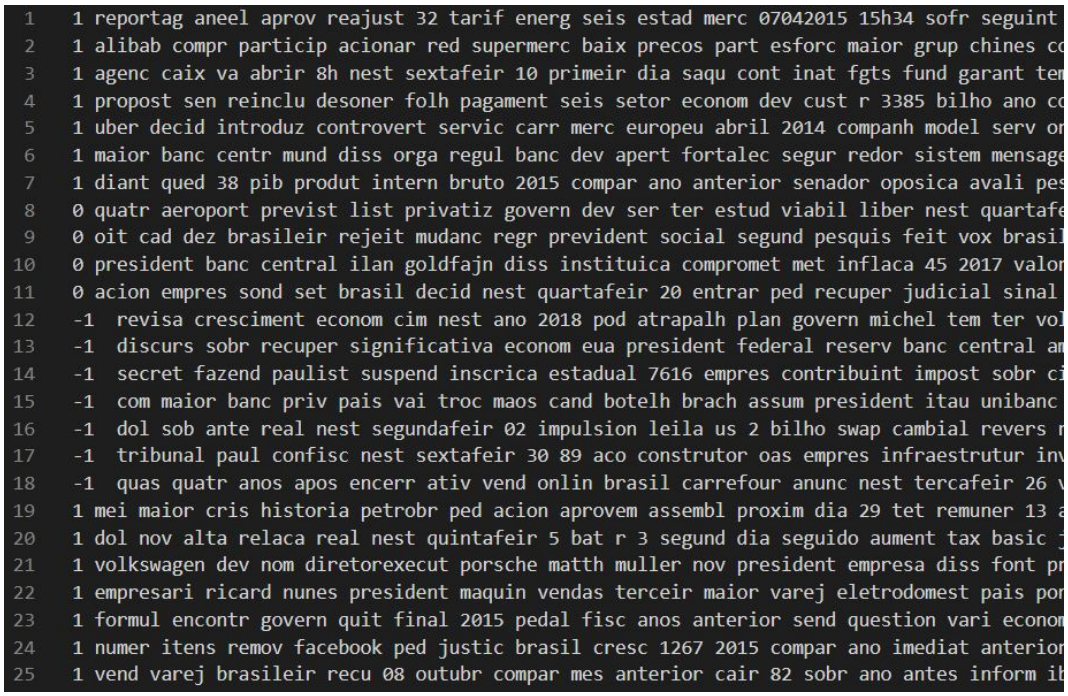


Figura 3.6: Exemplo do conjunto de dados criado para o ativo ABEV3 em uma janela de tempo de 1 dia no formato TSV.

Como pode ser visto na tabela 3.1, os registros da classe 1 e da classe -1 se mostram razoavelmente balanceados, porém, os registros da classe 0 se encontram em mínima quantidade em comparação com as outras duas classes, sendo até inexistente nas combinações ITUB4/2d e ITUB4/3d. Isto pode indicar a alta volatilidade do mercado de ações e a pequena possibilidade do preço de um ativo se manter o mesmo com o passar dos dias.

A possibilidade de utilizar técnicas de *Undersampling* foi descartada devida a mínima quantidade de registros da classe 0. Ao excluir registros da classe 1 e -1, os conjuntos de dados perderiam uma grande parte de informação, o que tornaria a tarefa de classificação textual inviável.

Testes envolvendo *Oversampling* foram feitos, mas os modelos encontrados demonstraram um desempenho pior do que testes feitos sem técnicas de balanceamento. Isto pode ter acontecido pelo fato de que, com o *Oversampling*, a informação de que a classe 0 tem uma menor possibilidade de aparecer nos dados é perdida. Sendo assim, de forma a avaliar o impacto que a classe 0 pode apresentar sobre o treinamento, os conjuntos de dados foram duplicados e os registros da classe 0 foram removidos das cópias. Assim como os conjuntos originais de 3 classes, os conjuntos de 2 classes também foram treinados e analisados. Com isso, o número final esperado de conjunto de dados foi de $25 \times 2 = 50$, porém, como as combinações ITUB4/2d e

Tabela 3.1: Quantidade de registros de cada classe para as combinações de ativo/período.

Conjunto	Classe 1	Classe 0	Classe -1
ABEV3/1d	10912	355	9679
ABEV3/2d	10955	235	9756
ABEV3/3d	10920	99	9927
ABEV3/4d	11081	160	9705
ABEV3/5d	11298	343	9305
BBDC4/1d	10579	79	10288
BBDC4/2d	10282	89	10575
BBDC4/3d	10605	24	10317
BBDC4/4d	11098	47	9801
BBDC4/5d	10954	139	9853
ITUB4/1d	10555	49	10342
ITUB4/2d	10253	0	10693
ITUB4/3d	10474	0	10472
ITUB4/4d	10712	54	10180
ITUB4/5d	10656	58	10232
PETR4/1d	10170	222	10554
PETR4/2d	10581	110	10255
PETR4/3d	10660	144	10142
PETR4/4d	11036	89	9821
PETR4/5d	10910	22	10014
VALE3/1d	10041	66	10839
VALE3/2d	10233	81	10632
VALE3/3d	10473	63	10410
VALE3/4d	10339	43	10564
VALE3/5d	10788	111	10047

ITUB4/3d não apresentaram a classe neutra, este número caiu para 48.

Seguindo o princípio de Pareto, todos os conjuntos foram divididos em subconjuntos de treino (80%) e teste (20%). Esta proporção foi aplicada separadamente sobre os registros de cada classe para evitar um balanceamento diferente do conjunto de dados original nos subconjuntos de treino e teste.

A rotina de manipulação, pré-processamento e criação dos conjuntos de dados está descrita no apêndice A.1. Foi feita utilizando a linguagem Python com o suporte de bibliotecas como pandas, numpy, csv, nltk e re.

Capítulo 4

Treinamento e Resultados

Como já dito anteriormente, o treinamento e a geração dos modelos preditivos foram feitos utilizando o framework PyText. Para isso, seria necessário escrever e executar 48 arquivos de configuração. Para evitar este esforço, uma rotina em Python foi desenvolvida para a criação dos arquivos de configuração e da execução destes pelo PyText para todos os conjuntos de dados. A rotina pode ser vista no apêndice A.2.

4.1 Configuração

Pela falta do arquivo de *embedding* pré-treinado, pela indisponibilidade de GPU na máquina utilizada e pela quantia de *epochs* empregada, cada treinamento de um conjunto de dados durava uma considerável quantia de tempo (entre 20 e 30 minutos, para uma rede CNN, e mais de 4 horas para uma rede BiLSTM), o que tornou a exploração de todos os parâmetros do PyText muito custosa. No entanto, foi possível realizar testes antes de definir uma configuração mais performática através da escolha do número de *epochs*, tipo de rede neural utilizada e *dropout*.

4.1.1 Número de *Epochs*

O número de *epochs* inicialmente decidido foi de 20, porém, testes mostraram que os modelos apresentavam um alto aumento no valor de custo a partir da 12ª *epoch*. Tornando razoável a decisão de usar 10 *epochs*, como o padrão definido pelo PyText.

Além disso, é notável a permanência da acurácia ou mesmo uma leve perda da mesma sobre o conjunto de teste a partir da 11ª *epoch*, possivelmente por conta do crescente sobre-ajuste sobre o conjunto de treino com o avanço das *epochs*.

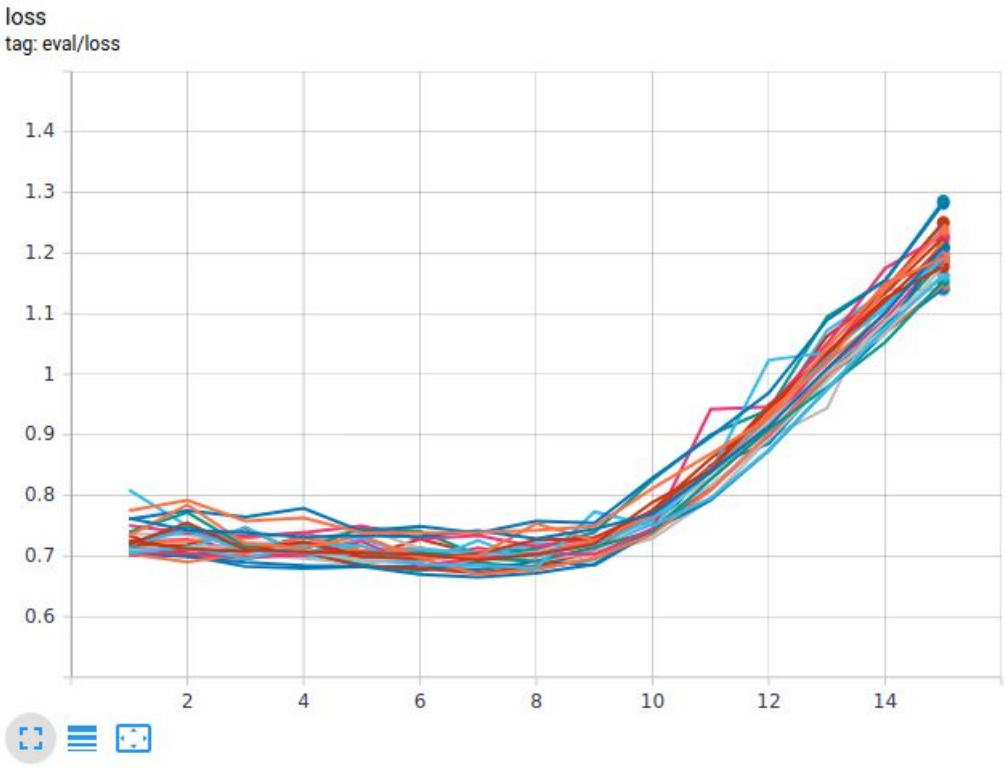


Figura 4.1: Gráfico de *loss* gerado pelo TensorBoard, indicando o aumento do custo a partir da 10ª *epoch* nos treinamentos feitos para os conjuntos de três classes.

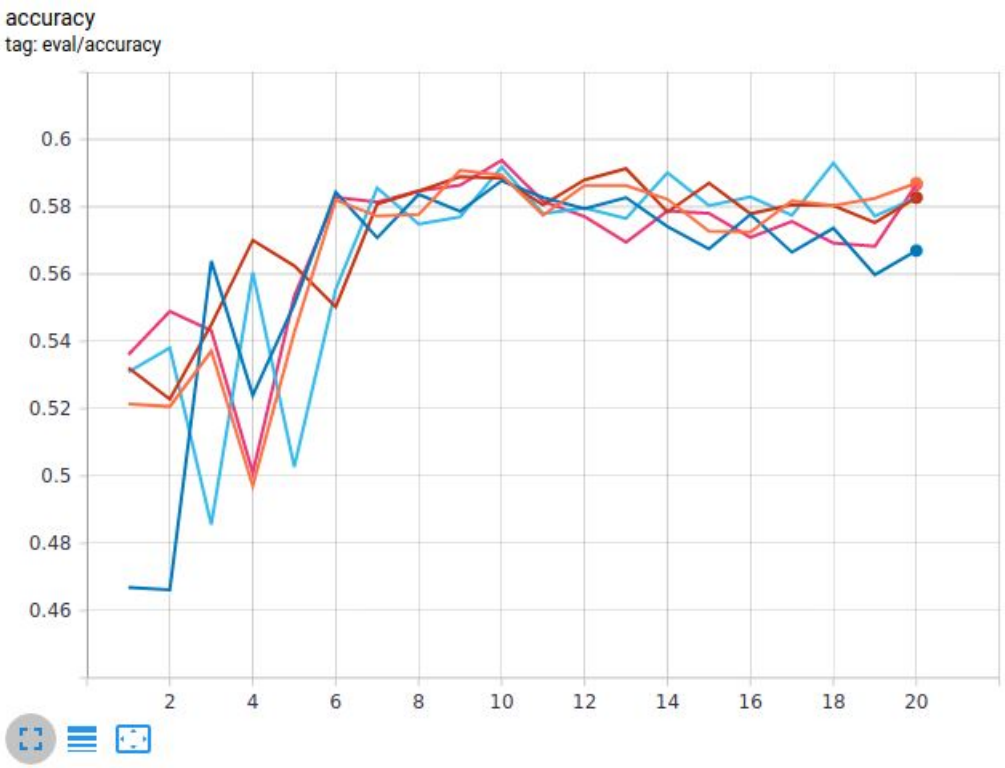


Figura 4.2: Gráfico de *accuracy* gerado pelo TensorBoard, indicando a acurácia em 20 *epochs* sobre os conjuntos dos ativos ABEV3.

Em alguns poucos casos, pode-se verificar um aumento de acurácia após a 10ª epoch. Logo, foi decidido um meio-termo entre a decisão inicial e o recomendado pelo PyText, optando assim por 15 *epochs*.

4.1.2 Tipo de Rede

Testes foram feitos com os dois tipos de redes neurais oferecidas pelo PyText para classificação de texto e mesmo com algumas alterações, a rede CNN obteve melhor performance em comparação a rede BiLSTM, tanto em acurácia quanto em tempo de processamento.

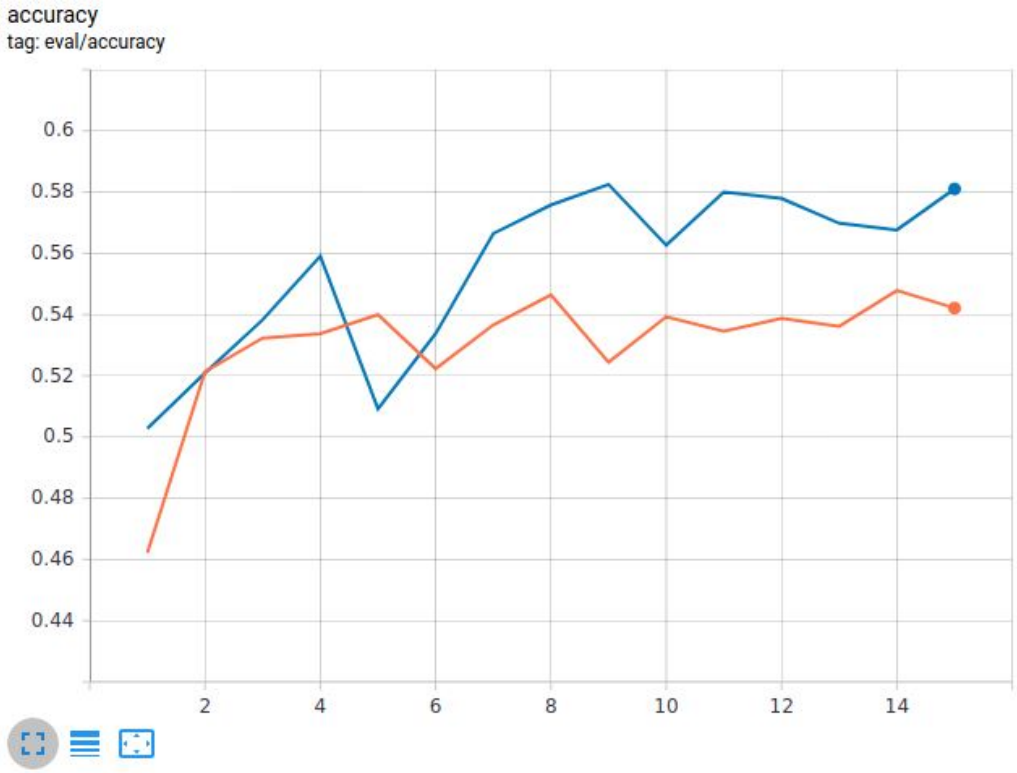


Figura 4.3: Gráfico de *accuracy* gerado pelo TensorBoard, indicando a acurácia sobre o conjunto de teste ao longo das *epochs*. Pode-se perceber que a rede CNN (curva azul) alcança um resultado melhor que rede BiLSTM (curva laranja) logo a partir da 4ª *epoch*. Teste realizado no conjunto ABEV3/1d.

Sendo assim, preferiu-se utilizar a rede CNN, o padrão utilizado pelo PyText quando o parâmetro de tipo de rede não é definido no arquivo de configuração.

4.1.3 *Dropout*

Foram testados os valores de *dropout* de $0,2$, $0,4$ e $0,6$, mas estes não demonstraram tanta diferença de performance entre si, tendo um pouco mais de vantagem

em relação a acurácia utilizando 0, 4, o padrão utilizado pelo PyText.

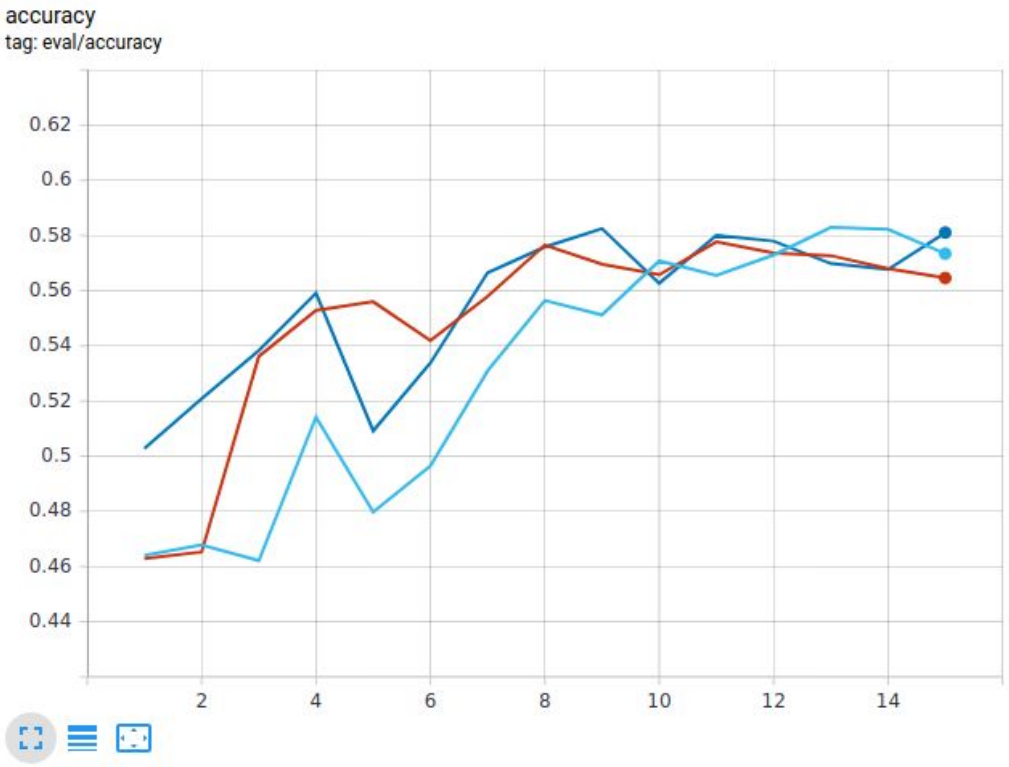


Figura 4.4: Gráfico de *accuracy* gerado pelo TensorBoard com a acurácia de três redes CNN para o conjunto ABEV3/1d: Com 0.2 de *dropout* (curva vermelha), 0.4 de *dropout* (curva azul escura) e 0.6 de *dropout* (curva azul clara).

4.2 Métricas e Análise

Tendo as definições de configuração do PyText prontas no modelo de arquivo de configuração (apêndice A.3), restou executar a rotina para a execução dos testes. Com isto, pode-se obter todas as métricas mencionadas em 2.2.3 e os arquivos exportados caffe2 de cada modelo preditivo.

Como mostrado na seção 3.3, os 48 conjuntos de dados foram treinados, resultando em modelos preditivos nesta mesma quantia. As métricas a seguir, associadas a estes modelos, está expressa em porcentagem (%).

4.2.1 Caso de 3 classes

Acurácia

- Maior acurácia: $60,21$ (BBDC4/4d)
- Menor acurácia: $55,43$ (VALE3/2d)
- Média de acurácia para ABEV3: $57,85$
- Média de acurácia para BBDC4: $58,60$
- Média de acurácia para ITUB4: $58,13$
- Média de acurácia para PETR4: $58,00$
- Média de acurácia para VALE3: $58,10$
- Média de acurácia para 1 dia: $57,79$
- Média de acurácia para 2 dias: $56,89$
- Média de acurácia para 3 dias: $57,86$
- Média de acurácia para 4 dias: $58,94$
- Média de acurácia para 5 dias: $58,91$
- Média de acurácia geral: $58,14$

Tabela 4.1: Acurácia para o caso de 3 classes.

Ativo / Dias	ABEV3	BBDC4	ITUB4	PETR4	VALE3
1	57,34	57,78	58,15	57,32	58,34
2	57,63	57,82	X	56,67	55,43
3	57,96	57,25	X	58,27	57,97
4	58,77	60,21	57,84	58,32	59,55
5	57,57	59,93	58,41	59,42	59,20

Precisão Média

- Maior precisão média: 72, 04 (PETR4/4d)
- Menor precisão média: 38, 28 (BBDC4/3d)
- Média de precisão média para ABEV3: 50, 43
- Média de precisão média para BBDC4: 39, 07
- Média de precisão média para ITUB4: 38, 82
- Média de precisão média para PETR4: 53, 10
- Média de precisão média para VALE3: 42, 10
- Média de precisão média para 1 dia: 41, 91
- Média de precisão média para 2 dias: 51, 88
- Média de precisão média para 3 dias: 41, 37
- Média de precisão média para 4 dias: 49, 19
- Média de precisão média para 5 dias: 42, 30
- Média de precisão média geral: 45, 22

Tabela 4.2: Precisão média para o caso de 3 classes.

Ativo / Dias	ABEV3	BBDC4	ITUB4	PETR4	VALE3
1	38,39	38,54	38,85	54,86	38,91
2	55,02	38,61	X	60,12	53,78
3	49,65	38,28	X	38,82	38,72
4	55,64	40,01	38,56	72,04	39,70
5	53,45	39,92	39,04	39,66	39,41

Cobertura Média

- Maior cobertura média: $41,29$ (ABEV3/5d)
- Menor cobertura média: $38,26$ (BBDC4/3d)
- Média de cobertura média para ABEV3: $40,28$
- Média de cobertura média para BBDC4: $39,06$
- Média de cobertura média para ITUB4: $38,86$
- Média de cobertura média para PETR4: $40,06$
- Média de cobertura média para VALE3: $39,23$
- Média de cobertura média para 1 dia: $39,06$
- Média de cobertura média para 2 dias: $39,63$
- Média de cobertura média para 3 dias: $39,10$
- Média de cobertura média para 4 dias: $40,00$
- Média de cobertura média para 5 dias: $39,90$
- Média de cobertura média geral: $39,55$

Tabela 4.3: Cobertura média para o caso de 3 classes.

Ativo / Dias	ABEV3	BBDC4	ITUB4	PETR4	VALE3
1	39,04	38,69	38,81	40,04	38,72
2	39,66	38,73	X	40,86	39,25
3	40,27	38,26	X	39,08	38,78
4	41,15	39,80	38,66	40,62	39,78
5	41,29	39,82	39,11	39,70	39,60

Medida F1

- Maior medida F1: 42,73 (PETR4/2d)
- Menor medida F1: 38,13 (BBDC4/3d)
- Média de medida F1 para ABEV3: 41,10
- Média de medida F1 para BBDC4: 38,91
- Média de medida F1 para ITUB4: 38,74
- Média de medida F1 para PETR4: 40,91
- Média de medida F1 para VALE3: 39,39
- Média de medida F1 para 1 dia: 39,05
- Média de medida F1 para 2 dias: 40,57
- Média de medida F1 para 3 dias: 39,28
- Média de medida F1 para 4 dias: 40,56
- Média de medida F1 para 5 dias: 40,07
- Média de medida F1 geral: 39,90

Tabela 4.4: Medida F1 média para o caso de 3 classes.

Ativo / Dias	ABEV3	BBDC4	ITUB4	PETR4	VALE3
1	38,55	38,59	38,63	41,07	38,43
2	40,28	38,64	X	42,73	40,62
3	41,39	38,13	X	38,94	38,64
4	42,58	39,67	38,61	42,18	39,74
5	42,71	39,53	38,97	39,63	39,50

4.2.2 Caso de 2 classes

Acurácia

- Maior acurácia: $60,52$ (BBDC4/5d)
- Menor acurácia: $56,12$ (PETR4/1d)
- Média de acurácia para ABEV3: $58,21$
- Média de acurácia para BBDC4: $58,48$
- Média de acurácia para ITUB4: $58,70$
- Média de acurácia para PETR4: $57,58$
- Média de acurácia para VALE3: $57,86$
- Média de acurácia para 1 dia: $57,29$
- Média de acurácia para 2 dias: $57,17$
- Média de acurácia para 3 dias: $58,30$
- Média de acurácia para 4 dias: $58,51$
- Média de acurácia para 5 dias: $59,56$
- Média de acurácia geral: $58,17$

Tabela 4.5: Acurácia para o caso de 2 classes.

Ativo / Dias	ABEV3	BBDC4	ITUB4	PETR4	VALE3
1	58,74	56,71	57,72	56,12	57,16
2	56,16	57,61	57,83	56,71	57,54
3	59,46	59,25	58,25	57,48	57,07
4	57,71	58,32	60,08	58,26	58,17
5	58,97	60,52	59,61	59,33	59,37

Precisão Média

- Maior precisão média: $60,62$ (BBDC4/5d)
- Menor precisão média: $56,10$ (PETR4/1d)
- Média de precisão média para ABEV3: $58,13$
- Média de precisão média para BBDC4: $58,49$
- Média de precisão média para ITUB4: $58,78$
- Média de precisão média para PETR4: $57,58$
- Média de precisão média para VALE3: $57,83$
- Média de precisão média para 1 dia: $57,35$
- Média de precisão média para 2 dias: $57,21$
- Média de precisão média para 3 dias: $58,30$
- Média de precisão média para 4 dias: $58,51$
- Média de precisão média para 5 dias: $59,44$
- Média de precisão média geral: $58,16$

Tabela 4.6: Precisão média para o caso de 2 classes.

Ativo / Dias	ABEV3	BBDC4	ITUB4	PETR4	VALE3
1	58,97	56,93	57,73	56,10	57,04
2	56,16	57,62	57,91	56,79	57,58
3	59,38	59,24	58,30	57,49	57,08
4	57,73	58,04	60,35	58,28	58,16
5	58,43	60,62	59,61	59,25	59,31

Cobertura Média

- Maior cobertura média: $60,64$ (BBDC4/5d)
- Menor cobertura média: $55,99$ (PETR4/1d)
- Média de cobertura média para ABEV3: $58,04$
- Média de cobertura média para BBDC4: $58,71$
- Média de cobertura média para ITUB4: $58,62$
- Média de cobertura média para PETR4: $57,56$
- Média de cobertura média para VALE3: $57,75$
- Média de cobertura média para 1 dia: $57,25$
- Média de cobertura média para 2 dias: $57,20$
- Média de cobertura média para 3 dias: $58,20$
- Média de cobertura média para 4 dias: $58,66$
- Média de cobertura média para 5 dias: $59,37$
- Média de cobertura média geral: $58,14$

Tabela 4.7: Cobertura média para o caso de 2 classes.

Dias \ Ativo	ABEV3	BBDC4	ITUB4	PETR4	VALE3
1	58,97	56,80	57,67	55,99	56,82
2	56,18	57,62	57,89	56,76	57,57
3	58,97	59,21	58,25	57,50	57,07
4	57,76	59,28	59,78	58,30	58,16
5	58,32	60,64	59,51	59,24	59,12

Medida F1

- Maior medida F1: $60,52$ (BBDC4/5d)
- Menor medida F1: $55,84$ (PETR4/1d)
- Média de medida F1 para ABEV3: $57,92$
- Média de medida F1 para BBDC4: $58,34$
- Média de medida F1 para ITUB4: $58,49$
- Média de medida F1 para PETR4: $57,49$
- Média de medida F1 para VALE3: $57,68$
- Média de medida F1 para 1 dia: $57,07$
- Média de medida F1 para 2 dias: $57,15$
- Média de medida F1 para 3 dias: $58,13$
- Média de medida F1 para 4 dias: $58,25$
- Média de medida F1 para 5 dias: $59,32$
- Média de medida F1 geral: $57,98$

Tabela 4.8: Medida F1 média para o caso de 2 classes.

Ativo / Dias	ABEV3	BBDC4	ITUB4	PETR4	VALE3
1	58,74	56,54	57,61	55,84	56,63
2	56,12	57,61	57,82	56,68	57,53
3	58,76	59,19	58,18	57,47	57,05
4	57,67	57,83	59,39	58,23	58,15
5	58,33	60,52	59,45	59,25	59,03

4.2.3 Análise

Pode-se verificar pelos dados acima que o melhor modelo encontrado, baseado em acurácia, foi o BBDC4/5d de duas classes. Entretanto, como erroneamente os registros da classe neutra não foram inseridos no conjunto de teste destes modelos, não podemos afirmar que este teria uma boa acurácia quando testado em dados reais. Sendo assim, foi considerado como melhor modelo o BBDC4/4d de 3 classes, com 60,21% de acurácia sobre o conjunto de teste.

Ainda assim de maneira geral, esta ainda é uma baixa taxa de acurácia, sendo ligeiramente melhor do que uma decisão aleatória entre duas classes (valorização ou desvalorização). Porém, se pensarmos que a classe neutra, mesmo com uma baixa possibilidade de aparição, é levada em conta então este valor de acurácia pode ser considerado um pouco melhor.

Com relação aos períodos de tempo, é notável que períodos maiores se mostraram com um desempenho melhor em comparação aos inferiores. Se tratando de acurácia, em média, os modelos preditivos de de 4 e 5 dias foram os que se mostraram mais performáticos. Isto pode indicar um efeito reverso de notícias sobre valorização de ativos onde é vista uma reação imediata na movimentação do mercado e após um período ocorre um movimento contrário a esta movimentação. Por exemplo, notícias ruins da AMBEV ocasionam uma queda instantânea nos ativos da empresa, mas por conta da sua estabilidade, o valor destes ativos voltam a subir logo depois

Considerando a volatilidade e incerteza do mercado, somado com a baixa disponibilidade de dados prontos de notícias sobre o mercado brasileiro, pode-se considerar razoável o desempenho do melhor modelo preditivo encontrado.

Capítulo 5

Considerações Finais

5.1 Conclusão

Diante da facilidade e praticidade do PyText, este trabalho demonstrou a possibilidade de criação de modelos preditivos voltados para o mercado financeiro de uma forma simples. Infelizmente, o PyText ainda não possui uma estrutura voltada para o treinamento paralelo de diferentes conjuntos de dados, o que resultou na necessidade da criação de scripts de criação e execução de configurações, isto seria interessante para a análise de diferentes modelos, como feito neste trabalho.

Ainda assim, o PyText se mostrou como uma boa e simples ferramenta para a criação de um modelo preditivo, principalmente pela facilidade de configuração através de um arquivo JSON. Sua funcionalidade de entrega e atualização de modelos em aplicações também é interessante para a manutenção de robôs de investimento que utilizem modelos preditivos, por exemplo.

Por fim, os resultados encontrados contribuem para o tema da criação de modelos preditivos com o intuito de ajudar operações de *Swing Trade*. Principalmente ao comparar e encontrar melhores períodos de tempo para segurar um determinado ativo com o intuito de lucrar com este baseado em um modelo preditivo.

5.2 Trabalhos Futuros

Como este trabalho apenas se baseou na análise dos modelos preditivos sobre dados de teste, seria interessante realizar testes com dados reais de forma progressiva. Isto seria possível ao desenvolver uma aplicação web que consumisse os modelos desenvolvidos no formato caffe2 e adquirisse notícias da rede através de APIs ou *web crawlers*, disponibilizando assim, o resultado dos modelos perante as notícias adquiridas em conjunto com a variação dos ativos em questão. Estes dados poderiam

ser salvos em um banco de dados para que posteriormente uma análise pudesse ser feita.

A utilização de ativos do iBovespa nos modelos preditivos gerados abre portas para a composição de um único modelo com o objetivo de prever variações do próprio iBovespa, isto seria interessante para a aplicação e saque de capital em fundos de investimento atrelados ao iBovespa, por exemplo.

A ferramenta PyText ainda pode ser mais explorada ao utilizar arquivos de *embeddings* pré-treinados em conjunto com a funcionalidade de utilizar uma GPU com CUDA para agilizar o processamento. Esta exploração poderia se basear na variação de parâmetros como quantidade e tamanho de *kernels* para uma rede CNN, por exemplo, ou mesmo a experimentação de novas arquiteturas disponibilizadas pelo PyText com o avanço de versão. O uso de novas ferramentas mais abertas e flexíveis também pode ser testado com o objetivo de comparar os resultados obtidos neste trabalho.

A aplicação de *online machine learning*, isto é, a aquisição de novos dados e treinamento automático com o intuito de melhorar um modelo em produção pode ser bem interessante para o caso de análise de notícias do mercado financeiro. Isto traria a possibilidade de fazer com que o modelo preditivo aprenda novos assuntos, palavras e frases da atualidade que podem estar impactando ativos financeiros como nunca antes.

Novas análises poderiam ser realizadas para contribuir para as informações dos modelos encontrados, como testes para verificar a latência e o *throughput* do modelo em caffe2. Como a performance foi definida apenas pela acurácia do modelo, seria também interessante avaliar o lucro auferido em cenários de teste e verificar a relação do lucro com a acurácia e precisão de cada classe.

Por fim, outras variações na construção do modelo podem ser adicionadas, como a inclusão de um limiar de oscilação para definir as classes 1 e -1 ou incorporar as amostras da classe 0 na classe -1, devido aos gastos operacionais que ocorrem quando não há lucro. O uso da probabilidade na saída do modelo também pode ser incorporado para auxiliar a decisão de *trading*, removendo a função *softmax* na camada de saída da rede, por exemplo.

Referências Bibliográficas

[1] LIU, Z., ZHU, H., CHONG, T. Y. "An NLP-PCA Based Trading Strategy On Chinese Stock Market", *Advances in Social Science and Education and Humanities Research*, v. 334, n. 2, pp. 80–89, jul. 2019.

[2] SEDLAK, M. "How Natural Language Processing is transforming the financial industry". https://www.ibm.com/blogs/watson/2016/06/natural-language-processing-transforming-financial-industry-2/, 2016. Acessado em Dezembro/2019.

[3] SMITH, D. J. "Efficient-Market Hypothesis (EMH) and Random-Walk Theory". https://stockmarketsupertrader.com/theory/efficient-market-hypothesis-emh-and-random-walk-theory/, 2018. Acessado em Dezembro/2019.

[4] VAZ, A. L. "Como lidar com dados desbalanceados em problemas de classificação". https://medium.com/data-hackers/como-lidar-com-dados-desbalanceados-em-problemas-de-classifica%C3%A7%C3%A3o-17c4d4357ef9, 2019. Acessado em Dezembro/2019.

[5] OGNJANOVSKI, G. "Everything you need to know about Neural Networks and Backpropagation — Machine Learning Easy and Fun". https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a, 2019. Acessado em Dezembro/2019.

[6] PRATEEK, N. "Statistics is Freaking Hard: WTF is Activation function". https://towardsdatascience.com/statistics-is-freaking-hard-wtf-is-activation-function-df8342cdf292, 2017. Acessado em Dezembro/2019.

[7] DERTAT, A. "Applied Deep Learning - Part 4: Convolutional Neural Networks". https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2, 2017. Acessado em Dezembro/2019.

[8] OLAH, C. "Understanding LSTM Networks". http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2017. Acessado em Janeiro/2020.

[9] DONAHUE, J., HENDRICKS, L. A., ROHRBACH, M., et al. "Long-term Recurrent Convolutional Networks for Visual Recognition and Description", jun. 2015.

[10] CUI, Z., KE, R., PU, Z., et al. "Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction", .

[11] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", 2014.

[12] AMEISEN, E. "How to solve 90 of NLP problems: a step-by-step guide". https://blog.insightdatascience.com/how-to-solve-90-of-nlp-problems-a-step-by-step-guide-fda605278e4e, 2018. Acessado em Dezembro/2019.

[13] MIKOLOV, T., CHEN, K., CORRADO, G., et al. "Efficient Estimation of Word Representations in Vector Space", 2013.

[14] "GitHub: facebookresearch / pytext". https://github.com/facebookresearch/pytext, . Acessado em Janeiro/2020.

[15] "Índice Bovespa (Ibovespa)". http://www.bmfbovespa.com.br/pt_br/produtos/indices/indices-amplos/indice-ibovespa-ibovespa-composicao-da-carteira.htm. Acessado em Dezembro/2019.

[16] ALY, A., LAKHOTIA, K., ZHAO, S., et al. "PYTEXT: A SEAMLESS PATH FROM NLP RESEARCH TO PRODUCTION", dez. 2018.

[17] ALVES, V. A. "IDENTIFICAÇÃO DE GÊNERO EM LETRAS MUSICAIS UTILIZANDO REDES PROFUNDAS E PYTEXT", jul. 2019.

[18] DO PAVINI, A. "Cresce número de pessoas físicas como profissionais na Bolsa". https://exame.abril.com.br/seu-dinheiro/cresce-numero-de-pessoas-fisicas-como-profissionais-na-bolsa/, 2019. Acessado em Dezembro/2019.

[19] BACHINSKIY, A. "The Growing Impact of AI in Financial Services: Six Examples". https://towardsdatascience.com/the-growing-impact-of-ai-in-financial-services-six-examples-da386c0301b2, 2019. Acessado em Dezembro/2019.

[20] CANTO, L. G. "Stock Market Predictor". https://github.com/lgcanto/stock-market-predictor/, 2019. Acessado em Dezembro/2019.

[21] MOREIRA, M. "Fusão entre BM&FBovespa e Cetip cria a B3, 5ª maior bolsa de valores do mundo". http://agenciabrasil.ebc.com.br/economia/noticia/2017-03/fusao-entre-bmfbovespa-e-cetip-cria-b3-5a-maior-bolsa-de-valores-do-mundo, 2017. Acessado em Dezembro/2019.

[22] WAWRZENIAK, D. "O Que É Análise Técnica?" https://www.bussoladoinvestidor.com.br/o-que-e-analise-tecnica/, 2018. Acessado em Dezembro/2019.

[23] WAWRZENIAK, D. "O Que É Análise Fundamentalista?" https://www.bussoladoinvestidor.com.br/o-que-e-analise-fundamentalista/, 2018. Acessado em Dezembro/2019.

[24] MALKIEL, B. G. "Efficient Market Hypothesis", *Finance*, pp. 127–134, 1989.

[25] "Análise Técnica x Análise Fundamentalista". https://www.tororadar.com.br/investimento/analise-tecnica/analise-tecnica-x-fundamentalista. Acessado em Dezembro/2019.

[26] JORDAN, M. I., MITCHELL, T. M. "Machine learning: Trends, perspectives, and prospects", *Science*, v. 349, pp. 80–89, jul. 2015.

[27] MITCHELL, T. M. *Machine Learning*. McGraw Hill, 1997.

[28] GULIPALLI, P. "The Pareto Principle for Data Scientists". https://www.kdnuggets.com/2019/03/pareto-principle-data-scientists.html, 2019. Acessado em Dezembro/2019.

[29] KOHAVI, R. "A Study of CrossValidation and Bootstrap for Accuracy Estimation and Model Selection", *AInternational joint Conference on artificial intelligence*, v. 14, pp. 1137––1145, 1995.

[30] GROVER, P. "5 Regression Loss Functions All Machine Learners Should Know". https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0, 2018. Acessado em Dezembro/2019.

[31] ROJAS, R. *Neural Networks - A Systematic Introduction*. Springer, 1996.

[32] SAK, H., SENIOR, A., BEAUFAYS, F. "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling", 2014.

[33] AMIDI, S. "Recurrent Neural Networks cheatsheet". https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks. Acessado em Janeiro/2020.

[34] HOCHREITER, S., SCHMIDHUBER, J. "Long Short-Term Memory", *Neural Computation*, jun. 1997.

[35] GARBADE, M. J. "A Simple Introduction to Natural Language Processing". https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32, 2018. Acessado em Dezembro/2019.

[36] "Repositório de Word Embeddings do NILC". http://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc, 2017. Acessado em Dezembro/2019.

[37] "PyText Documentation". https://pytext.readthedocs.io/en/master/index.html, 2018. Acessado em Janeiro/2020.

[38] "PyTorch". https://pytorch.org/, . Acessado em Janeiro/2020.

[39] "Folha de S. Paulo". https://www.folha.uol.com.br/. Acessado em Dezembro/2019.

[40] "InfoMoney". https://www.infomoney.com.br/. Acessado em Dezembro/2019.

[41] "Twitter API Reference - Search Tweets". https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets. Acessado em Dezembro/2019.

[42] MARLESSON. "Kaggle - News of the Brazilian Newspaper". https://www.kaggle.com/marlesson/news-of-the-site-folhauol, 2019. Acessado em Dezembro/2019.

[43] "B3 - Séries históricas". http://www.bmfbovespa.com.br/pt_br/servicos/market-data/historico/mercado-a-vista/series-historicas/. Acessado em Dezembro/2019.

[44] CAMPEAO, D. "Kaggle - Bovespa". https://www.kaggle.com/dcampeao/bovespa, 2018. Acessado em Dezembro/2019.

Apêndice A

Rotinas e Arquivos de Configuração

A.1 Rotina de Pré-processamento e Criação de Conjuntos de Dados

```
import re
import numpy as np
import pandas as pd
import csv
import os
from datetime import timedelta
from unicodedata import normalize
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.snowball import SnowballStemmer

OUT_DIR = './dataset_out'
MAX_DAYS = 5
TRAIN_PERCENTAGE_SIZE = 80/100
TEST_PERCENTAGE_SIZE = 20/100
REGEXP_REMOVE_SPECIAL = re.compile('[^a-zA-Z0-9 ]+')
ONLY_ONE_CODE = False
ONLY_ONE_CODE_NAME = 'VALE3'
STOPWORDS = stopwords.words('portuguese')
STEMMER = SnowballStemmer('portuguese')
ARTIGOS_TEXT_COLUMN = 'text'
ARTIGOS_UNUSED_COLUMNS = ['title', 'category', 'subcategory', 'link']
BOVESPA_UNUSED_COLUMNS = ['open', 'company', 'typereg', 'bdicode',
↪  'markettype', 'spec', 'prazot', 'currency', 'max', 'min', 'med',
↪  'preofc', 'preofv', 'totneg', 'quatot']
```

```
25
26 df_companies = pd.read_csv('../datasets/company-codes.csv')
27 df_bovespa = pd.read_csv('../datasets/kaggle/bovespa.csv')
28 df_articles = pd.read_csv('../datasets/kaggle/articles.csv')
29
30 def getAppreciation(before, after):
31   if after > before:
32     return 1
33   if before > after:
34     return -1
35   else:
36     return 0
37
38 def getEffectDate(date):
39   effectDate = date
40   while (df_bovespa[df_bovespa.date == effectDate].size < 1):
41     effectDate = effectDate + timedelta(days=1)
42   return effectDate
43
44 def exportToTSV(dataframe, filename):
45   if not os.path.exists(OUT_DIR):
46     os.mkdir(OUT_DIR)
47   fullpath = '%s/%s' % (OUT_DIR, filename)
48   dataframe.to_csv(fullpath, sep='\t', quoting=csv.QUOTE_NONE,
   ↪  index=False, header=False)
49
50 def getCleanText(text):
51   finalTextArray = []
52   lowerText = text.lower()
53   for word in lowerText.split():
54     if word not in STOPWORDS:
55       finalTextArray.append(STEMMER.stem(word))
56   finalText = ' '.join(finalTextArray)
57   finalText = normalize('NFKD', finalText).encode('ASCII',
   ↪  'ignore').decode('ASCII')
58   finalText = REGEXP_REMOVE_SPECIAL.sub('', finalText)
59   finalText = re.sub(' +', ' ', finalText)
60   return finalText
61
62 df_articles = df_articles[df_articles.category == 'mercado']
63 df_articles.drop(ARTIGOS_UNUSED_COLUMNS, inplace=True, axis=1)
```

```
df_articles['date'] = df_articles.apply(lambda row:
↪  np.int64(row['date'].replace('-', '')), axis=1)
df_articles['date'] = pd.to_datetime(df_articles['date'].astype(str),
↪  format='%Y%m%d')

df_bovespa.columns = map(str.lower, df_bovespa.columns)
df_bovespa =
↪  df_bovespa[df_bovespa.codneg.str.strip().isin(df_companies.code)]
df_bovespa['date'] = pd.to_datetime(df_bovespa['date'].astype(str),
↪  format='%Y%m%d')
df_bovespa = df_bovespa[df_bovespa.date >= df_articles.date.min()]
df_bovespa.drop(BOVESPA_UNUSED_COLUMNS, inplace=True, axis=1)
df_bovespa = df_bovespa.sort_values('date')

df_articles['date'] = df_articles.apply(lambda row:
↪  getEffectDate(row.date), axis=1)
df_articles[ARTIGOS_TEXT_COLUMN] = df_articles.apply(lambda row:
↪  getCleanText(row[ARTIGOS_TEXT_COLUMN]), axis=1)
df_articles = df_articles.sort_values('date')

df_analysis = pd.DataFrame(columns=['dataset','1s','0s', '-1s'])

for index, row in df_companies.iterrows():
  if not ONLY_ONE_CODE or (ONLY_ONE_CODE and row['code'] ==
  ↪  ONLY_ONE_CODE_NAME):
    print('Generating for ' + row['code'])
    df_full = df_bovespa[df_bovespa.codneg.str.strip() == row['code']]
    df_full =
    ↪  df_full.assign(close_before=df_full['close'].transform(lambda
    ↪  group: group.shift(1)))
    df_full = df_full[~np.isnan(df_full.close_before)]
    for d in range(MAX_DAYS):
      interval = d + 1
      df_interval =
      ↪  df_full.assign(close_after=df_full['close'].transform(lambda
      ↪  group: group.shift(-interval)))
      df_interval = df_interval[~np.isnan(df_interval.close_after)]
      df_interval.drop('close', inplace=True, axis=1)
      df_company = pd.merge(df_articles, df_interval, on='date',
      ↪  how='inner')
      if df_company.size > 0:
```

```
93          df_company.drop(['date', 'codneg'], inplace=True, axis=1)
94          df_company['label'] = df_company.apply(lambda row:
            ↪  getAppreciation(row['close_before'], row['close_after']),
            ↪  axis=1)
95          df_company.drop(['close_before', 'close_after'], inplace=True,
            ↪  axis=1)
96          df_company = df_company[['label', ARTIGOS_TEXT_COLUMN]]
97
98          df_company_positive = df_company[df_company.label ==
            ↪  1].sample(frac=1)
99          df_company_neutral = df_company[df_company.label ==
            ↪  0].sample(frac=1)
100         df_company_negative = df_company[df_company.label ==
            ↪  -1].sample(frac=1)
101
102         analysis = pd.Series({"dataset": row['code'] + '_' + str(interval)
            ↪  + 'd.tsv', "1s": len(df_company_positive), "0s":
            ↪  len(df_company_neutral), "-1s": len(df_company_negative)})
103         df_analysis = df_analysis.append(analysis, ignore_index=True)
104
105         trainPositiveSize =
            ↪  round(len(df_company_positive)*(TRAIN_PERCENTAGE_SIZE))
106         testPositiveSize =
            ↪  round(len(df_company_positive)*(TEST_PERCENTAGE_SIZE))
107
108         trainNeutralSize =
            ↪  round(len(df_company_neutral)*(TRAIN_PERCENTAGE_SIZE))
109         testNeutralSize =
            ↪  round(len(df_company_neutral)*(TEST_PERCENTAGE_SIZE))
110
111         trainNegativeSize =
            ↪  round(len(df_company_negative)*(TRAIN_PERCENTAGE_SIZE))
112         testNegativeSize =
            ↪  round(len(df_company_negative)*(TEST_PERCENTAGE_SIZE))
113
114         df_company_train = df_company_positive.head(trainPositiveSize)
115         df_company_positive = df_company_positive.iloc[trainPositiveSize:]
116         df_company_train =
            ↪  df_company_train.append(df_company_negative.head(trainNegativeSize))
117         df_company_negative = df_company_negative.iloc[trainNegativeSize:]
118
```

```
        df_company_test = df_company_positive.head(testPositiveSize)
        df_company_positive = df_company_positive.iloc[testPositiveSize:]
        df_company_test =
        ↪  df_company_test.append(df_company_negative.head(testNegativeSize))
        df_company_negative = df_company_negative.iloc[testNegativeSize:]

        exportToTSV(df_company_train, row['code'] + '_2c_' + str(interval)
        ↪  + 'd_train.tsv')
        exportToTSV(df_company_test, row['code'] + '_2c_' + str(interval)
        ↪  + 'd_test.tsv')

        df_company_train =
        ↪  df_company_train.append(df_company_neutral.head(trainNeutralSize))
        df_company_neutral = df_company_neutral.iloc[trainNeutralSize:]
        df_company_test =
        ↪  df_company_test.append(df_company_neutral.head(testNeutralSize))
        df_company_neutral = df_company_neutral.iloc[testNeutralSize:]

        exportToTSV(df_company_train, row['code'] + '_3c_' + str(interval)
        ↪  + 'd_train.tsv')
        exportToTSV(df_company_test, row['code'] + '_3c_' + str(interval)
        ↪  + 'd_test.tsv')
```

A.2 Rotina de Criação de Arquivos de Configuração e Execução de Treinamentos

```
import os
import pandas as pd
import subprocess
import glob

MAX_DAYS = 5
df_companies = pd.read_csv("../datasets/company-codes.csv")

for index, row in df_companies.iterrows():
  asset = row['code']
  for d in range(MAX_DAYS):
    interval = d + 1
    # configName = asset + "_2c_" + str(interval) + "d"
    configName = asset + "_3c_" + str(interval) + "d"
    fileName = "configs/" + configName + ".json"
```

```
    with open("templateconfig_XXXXX_Yc_Zd.json") as inputFile,
    ↪  open(fileName, "w") as outputFile:
      for line in inputFile:
        outputFile.write(line.replace("XXXXX_Yc_Zd", configName))

    print("Executing for " + configName)

    bashCommand = "pytext train < " + fileName
    result = subprocess.run(bashCommand, shell=True,
    ↪  stdout=subprocess.PIPE)
    result.stdout.decode('utf-8')

    print("Renaming run folder")

    runFoldersList = glob.glob("runs/*")
    latestRunFolder = max(runFoldersList, key=os.path.getctime)
    os.rename(latestRunFolder, "runs/" + configName)

    print("Execution done for " + configName)
```

A.3 Modelo de Arquivo de Configuração

```
{
  "version": 18,
  "task": {
    "DocumentClassificationTask": {
      "data": {
        "source": {
          "TSVDataSource": {
            "field_names": ["label", "text"],
            "train_filename":
            ↪  "../preprocessor/dataset_out/XXXXX_Yc_Zd_train.tsv",
            "test_filename":
            ↪  "../preprocessor/dataset_out/XXXXX_Yc_Zd_test.tsv",
            "eval_filename":
            ↪  "../preprocessor/dataset_out/XXXXX_Yc_Zd_test.tsv"
          }
        }
      },
      "model": {
        "DocModel": {
```

```
          "representation": {
            "DocNNRepresentation": {}
          }
        }
      },
      "trainer": {
        "epochs": 15
      },
      "metric_reporter": {
        "output_path": "metric_reports/XXXXX_Yc_Zd.txt",
        "model_select_metric": "accuracy",
        "target_label": null,
        "text_column_names": [
          "text"
        ]
      }
    }
  },
  "export_torchscript_path": "torchscripts/XXXXX_Yc_Zd.pt1",
  "export_caffe2_path": "caffe2_exports/XXXXX_Yc_Zd.caffe2.predictor"
}
```

A.4 Arquivo com Lista de Ativos a Serem Processados

```
code,names
ABEV3,AMBEV
BBDC4,BRADESCO
ITUB4,ITAU
PETR4,PETROBRAS
VALE3,Vale
```