REDES NEURAIS CONVOLUCIONAIS E MÁQUINAS DE APRENDIZADO EXTREMO APLICADAS AO MERCADO FINANCEIRO BRASILEIRO

Elisangela Lopes de Faria

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Civil, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Civil.

Orientadores: Nelson Francisco Favilla Ebecken
Marcelo Portes de Albuquerque

Rio de Janeiro
Setembro de 2018

REDES NEURAIS CONVOLUCIONAIS E MÁQUINAS DE APRENDIZADO EXTREMO APLICADAS AO MERCADO FINANCEIRO BRASILEIRO

Elisangela Lopes de Faria

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA CIVIL.

Examinada por:

_____
Prof. Nelson Francisco Favilla Ebecken, D.Sc.

_____
Prof. Marcelo Portes de Albuquerque, D.Sc.

_____
Prof. Jose Luis Drummond Alves, D.Sc.

_____
Prof. Mario Antonio Ribeiro Dantas, D.Sc.

_____
Prof. Elton Fernandes, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2018

Faria, Elisangela Lopes de

Redes Neurais Convolucionais e Máquinas de Aprendizado Extremo aplicadas ao mercado financeiro brasileiro/ Elisangela Lopes de Faria. – Rio de Janeiro: UFRJ/COPPE, 2018.

XIII, 135 p.: il.; 29,7 cm.

Orientadores: Nelson Francisco Favilla Ebecken

Marcelo Portes de Albuquerque

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia Civil, 2018.

Referências Bibliográficas: p. 120-135.

1. Redes Neurais de Aprendizado Profundo. 2. Word2vec. 3. Mercados Financeiros. I. Ebecken, Nelson Francisco Favilla *et al*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Civil. III. Título.

Dedico este trabalho à minha mãe Maria e ao meu pai José Lopes (*in memorian*) que me mostraram desde pequena o quanto era importante estudar, mesmo sem nunca terem tido a oportunidade de frequentarem uma escola.

AGRADECIMENTOS

A Deus, pela oportunidade de poder vivenciar este momento tão importante na conclusão de mais uma jornada.

Aos meus pais, exemplos de coragem, trabalho, honestidade e determinação. Pai, o senhor não está mais presente fisicamente, mas seu carinho estará sempre em meu coração e seus ensinamentos me acompanharão por toda a minha vida.

Ao meu marido Jorge Luis, meu eterno agradecimento pelas palavras de carinho e incentivo e por compreender todos os momentos de dificuldade que vivenciamos juntos durante esta fase de nossas vidas.

De uma maneira muito especial, ao meu filho Jorge Gabriel, que veio ao mundo durante esta fase de minha vida e mesmo não tendo idade pra entender o que é uma tese, agradeço pelo carinho e amor incondicional, que sempre me estimularam nos momentos difíceis. E me desculpe pelas vezes que não pude atendê-lo, pois precisava estudar.

À minha família, em especial minha irmã Marlene, minhas sobrinhas Jane e Cyntia e meu cunhado Ronaldo pelo carinho, apoio, torcida e confiança que sempre depositaram em mim.

Aos professores da Coppe, em particular a Beatriz de Souza Leite Pires de Lima e Alexandre Gonçalves Evsukoff e de uma maneira muito especial ao meu orientador Nelson Francisco Favilla Ebecken por todos os ensinamentos transmitidos, pela contribuição no meu processo de aprendizado e pela confiança e apoio durante o desenvolvimento deste trabalho.

Ao prof. Marcelo Portes de Albuquerque pela oportunidade de tê-lo como orientador desde a Iniciação Científica. Tenho orgulho de estar todos estes anos sob sua supervisão no Centro Brasileiro de Pesquisas Físicas (CBPF). Agradeço pela confiança, conselhos, paciência, amizade e principalmente por sua contribuição em minha formação acadêmica e profissional.

Ao Prof. Marcio Portes de Albuquerque pela contribuição dada em minha formação acadêmica.

Ao colega da Coppe Ivo Gersberg pelas conversas e discussões sobre o doutorado e por sua ajuda durante o desenvolvimento deste trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

REDES NEURAIS CONVOLUCIONAIS E MÁQUINAS DE APRENDIZADO EXTREMO APLICADAS AO MERCADO FINANCEIRO BRASILEIRO

Elisangela Lopes de Faria

Setembro/2018

Orientadores: Nelson Francisco Favilla Ebecken
Marcelo Portes de Albuquerque

Programa: Engenharia Civil

Este trabalho apresenta um estudo preditivo do principal índice do mercado acionário brasileiro (Ibovespa) utilizando informações econômicas e financeiras extraídas de importantes provedores de notícias financeiras do Brasil. Um modelo híbrido é proposto combinando as potencialidades de duas arquiteturas de redes neurais artificiais distintas, uma rede neural Convolucional (arquitetura de aprendizado profundo) e uma máquina de aprendizado extremo, juntamente com técnicas de Processamento de Linguagem Natural, mais precisamente as técnicas de representação distribuída das palavras (*word embeddings*). A influência nos resultados dos principais parâmetros intrínsecos ao modelo proposto é mostrada e discutida. A acurácia máxima obtida com o modelo preditivo foi de 60,2%. Uma estratégia de negociação foi desenvolvida conforme as previsões do modelo reportando lucratividade superior quando comparada com a estratégia *buy and hold*. Por fim, os resultados obtidos sobre este mercado emergente são equivalentes a outros similares reportados na literatura, porém em mercados bem desenvolvidos.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

CONVOLUTION NEURAL NETWORK AND EXTREME LEARNING MACHINE APPLIED TO THE BRAZILIAN FINANCIAL MARKET

Elisangela Lopes de Faria

September/2018

Advisors: Nelson Francisco Favilla Ebecken
Marcelo Portes de Albuquerque

Department: Civil Engineering

This work presents a predictive study of the main index of the Brazilian stock market (Ibovespa) by using economic and financial information extracted from important financial news providers in Brazil. A hybrid model is proposed combining the potentialities of two distinct artificial neural network architectures, a convolutional neural network (deep learning architecture) and an extreme learning machine, together with techniques of Natural Language Processing, more precisely techniques of distributed representation of words (word embeddings). The influence on the results of the different parameters intrinsic to the proposed model is show and discussed. The maximum accuracy obtained with the predictive model was 60,2%. A trading strategy was developed according to the model predictions, reporting superior profitability as compared to the buy and hold strategy. Finally, the results obtained on this emerging market are similar to the other found in studies reported in the literature, although performed on well developed markets.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

RNA	Redes Neurais Artificiais
DLNN	*Deep Learning Neural Network*
MLP	*Multilayer Perceptrons*
SVM	*Support Vector Machine*
TI	Tecnologia da Informação
RNTN	*Recursive Neural Tensor Network*
ELM	*Extreme Learning Machine*
CNN	*Convolution Neural Network*
PLN	Processamento de Linguagem Natural
IA	Inteligência Artificial
MCP	Modelo de *McCulloch* e *Pitts*
BOW	*Bag of words*
PMI	*Pointwise Mutual Information*
CPU	*Central Processing Unit*
GPU	*Graphics Processing Unit*
NLTK	*Natural Language Toolkit*
B3	Bolsa de valores oficial do Brasil, sediada na cidade de São Paulo
S&P500	*Standard & Poor's 500*
NASDAQ	*National Association of Securities Dealers Automated Quotations*

1.Introdução

Atualmente os mercados financeiros vêm desempenhando um papel muito importante na organização social e econômica dos países. Em aplicações envolvendo estes tipos de mercados, a informação pode ser considerada como algo muito valioso. O avanço da computação e das comunicações nos dias atuais aperfeiçoou os mercados financeiros tornando-os mais eficientes, mas ao mesmo tempo fez com que o volume de informação aumentasse vertiginosamente dificultando assim a tarefa dos investidores em suas tomadas de decisões.

Nos últimos anos, diferentes estudos e teorias foram desenvolvidos para tentar explicar o comportamento caótico dos diferentes mercados financeiros mundiais. Duas abordagens comumente utilizadas nestes estudos estão relacionadas a análise técnica e a análise fundamentalista. A primeira abordagem tenta modelar o comportamento histórico de um ativo financeiro explorando implicações escondidas em atividades de negociação do ativo no passado, através de análise de padrões e tendências existentes em dados de séries temporais do mercado financeiro. Já a segunda abordagem investiga os fatores econômicos que podem afetar o movimento do mercado, como por exemplo, indicadores de desempenho de uma empresa, sua estratégia de negócios, política administrativa, solidez financeira, entre outros.

Na busca por métodos eficientes que possam entender o elevado grau de complexidade destes sistemas, assim como a não linearidade de suas variáveis, muitas pesquisas, motivadas pelas abordagens acima foram realizadas com o uso de técnicas de aprendizado de máquina, mais precisamente as Redes Neurais Artificiais (RNAs) (HAYKIN, 2001).

Dentro do contexto da análise técnica, dados históricos das ações ou mesmo indicadores econômicos podem ser utilizados sem quaisquer suposições estatísticas (FARIA *et al*., 2009) como dados de entrada das RNAs para tentar entender ou mesmo predizer o comportamento futuro do mercado. Por outro lado, na análise fundamentalista, notícias financeiras (SCHUMAKER *et al*., 2009), relatórios financeiros (WANG *et al*., 2012) e até mesmo informações em micro blogs (RUIZ *et al*., 2012) são consideradas importantes fontes de informações para prever o comportamento futuro do mercado juntamente com as técnicas de RNAs.

A nova geração de RNAs denominadas Redes Neurais de Aprendizado Profundo (*Deep Learning Neural Network* - DLNN) que tem como base modelos de arquitetura com

várias camadas de processamento (HINTON *et al*., 2006 ; BENGIO *et al*., 2007), permitiu uma maior capacidade de aprendizagem e generalização e um desempenho excelente em muitos problemas importantes como na área de visão computacional (KRIZHEVSKY *et al*., 2012), em reconhecimento de fala (GRAVES *et al*., 2013) e caracteres (HINTON *et al*., 2006) e reconhecimento facial (FAN *et al*., 2014).

O que torna as DLNNs tão atrativas nos dias atuais é o fato das mesmas conseguirem resolver uma série de problemas que limitavam a expansão e a implementação bem sucedida dos modelos considerados como arquiteturas rasas (do inglês *shallow architetures*), que são as arquiteturas de modelos, como por exemplo, as Redes Neurais Múltiplos Perceptrons (MLP) (HAYKIN, 2001) e Máquina de Vetor de Suporte (S*upport Vector Machine* - SVM) (VAPNIK, 1998). Muitas destas limitações, segundo (BENGIO *et al*., 2007) estão relacionadas a três características básicas que devem fazer parte de um algoritmo de aprendizado de máquina, a saber: (i) um menor custo computacional possível; (ii) a utilização de menos padrões de exemplo para uma boa generalização; (iii) e por fim um menor envolvimento humano na construção do modelo antes do treinamento. Os autores afirmaram também que as redes DLNNs tentam de alguma forma suprir as limitações das arquiteturas rasas por apresentarem tais características em seus modelos.

Outro ponto a ser destacado é que embora um número de teoremas possa mostrar que certos tipos de arquiteturas rasas (MLP, SVM, etc.) consigam aproximar qualquer tipo de função com precisão arbitrária, os mesmos não são capazes de inferir a eficiência desta representação. Entretanto as DLNNs mesmo não conseguindo apresentar funções de custo convexa, conseguem a princípio representar certas famílias de funções de forma mais eficiente do que as arquiteturas rasas (BENGIO *et al*., 2007).

As redes DLNNs também se diferenciam das arquiteturas rasas por apresentarem mais camadas de processamento (várias camadas escondidas). De uma maneira geral, (BENGIO *et al*., 2007) define uma DLNN como uma composição de várias camadas de componentes parametrizados e não lineares, sendo estes parâmetros aprendidos ao longo do treinamento da rede.

Atualmente, os métodos industriais mais avançados da visão computacional e de reconhecimento de voz são baseados em redes DLNNs. Gigantes da indústria de Tecnologia da Informação (TI) como *Apple*, *Google* e *Facebook* estão empregando pesquisadores em desenvolvimento para este tipo de arquitetura profunda.

Dado o grande sucesso das DLNNs, surgem perspectivas de utilização das mesmas em outras linhas de pesquisa, como é o caso da previsão do comportamento do mercado

financeiro, utilizando informação não estruturada (dados textuais) e Processamento de Linguagem Natural (PLN).

FEHRER *et al*. (2015) treinaram um modelo DLNN para prever o retorno das ações do mercado financeiro alemão com base em *headlines* (títulos) de notícias em inglês. Os autores usaram uma rede DLNN *autoencoder* recursiva para estimar probabilidades. Os dados foram classificados em três classes (cima, baixo ou estável) e o objetivo dos autores foi prever o valor do retorno do preço de ações um dia a frente. Nesse trabalho foi reportado que as redes utilizadas conseguiram superar os resultados de um modelo randômico em 5.66% de acurácia. DING *et al*. (2015) também fizeram uso de uma rede DLNN com base em *headlines* extraídos da *Reuters* e *Bloomberg* para tentar prever o retorno diário do *S&P*500 (*Standard* & *Poor's* 500). Nesse trabalho primeiramente os eventos foram extraídos a partir dos textos, representados como vetor e treinados por uma rede *Recursive Neural Tensor Network* (RNTN). Em seguida usando uma outra arquitetura de DLNN, a Rede Neural Convolucional (*Convolutional Neural Network* - CNN), os autores estimaram a influência de eventos de longo e curto prazo nos preços de diferentes ações que fazem parte do *S&P*500. Os autores reportaram uma melhoria de 6% de acurácia nos resultados obtidos quando comparados a métodos tradicionais utilizados na literatura. Além disto, os autores afirmaram também que os resultados obtidos em simulação do mercado tiveram uma lucratividade maior do que os sistemas de negociação existentes na *S&P*500 e baseados em dados históricos das ações.

1.1 Objetivos do Estudo

Conforme citado anteriormente, apesar de algumas tentativas de estudar o comportamento de alguns mercados financeiros utilizando dados textuais e DLNN, esta ainda é uma área emergente de pesquisa com resultados tímidos e pouco expressivos, devido à dificuldade de se entender a complexidade destes sistemas altamente dinâmicos e não lineares. Dentro deste contexto a atual proposta desta tese visa o aprofundamento, investigação das potencialidades e uso de duas arquiteturas distintas de redes neurais, a rede DLNN, mais precisamente a CNN e a rede Máquina de Aprendizado Extremo (*Extreme Learning Machine* - ELM), juntamente com técnicas de Processamento de Linguagem Natural para quantificar ou predizer a tendência diária do comportamento futuro do índice (Ibovespa) do mercado acionário brasileiro (BmfBovespa – Bolsa de Valores, Mercadorias e Futuros). Os objetivos propostos para esta tese podem ser divididos em quatro etapas:

- Analisar e investigar as potencialidades do uso da rede neural CNN no processamento de dados textuais financeiros;
- Avaliar o uso da representação distribuída dos documentos para a extração de informações a partir de dados não estruturados escritas na língua portuguesa;
- Avaliar o uso das redes ELM como uma alternativa ao processo de classificação das características obtidas a partir da rede neural CNN.
- Verificar e analisar a capacidade preditiva do modelo proposto através de métricas de avaliação citadas na literatura.

A proposta desta tese aponta então para o entendimento e uso das redes neurais DLNNs, mais precisamente uma arquitetura específica denominada CNN (que faz parte da última geração de redes neurais) e a rede ELM no processamento de dados textuais financeiros. As redes neurais propostas, em especial a CNN são motivadas pelo fato de serem técnicas desenvolvidas para a área de visão computacional, em especial no campo de reconhecimento de imagens (HINTON *et al*., 2006), mas com uma possível potencialidade para uso em diferentes aplicações envolvendo PLN como por exemplo em análise semântica (YIH *et al*., 2014), recuperação de informação (SHEN *et al*., 2014), modelagem de sentenças (KALCHBRENNER *et al*., 2014) e em aplicações tradicionais (COLLOBERT *et al*., 2011).

Por se tratar de uma área muito recente, muitos pontos ainda estão em aberto. A ideia é discutir neste trabalho como as principais características da CNN podem ser exploradas dentro do contexto de aplicações de PLN. Por exemplo, invariância a localização e composicionalidade local, duas das principais características da CNN, podem ser facilmente identificadas em computação visual através dos pixels que representam a imagem (pixels próximos uns dos outros são suscetíveis de estarem semanticamente correlacionados, independentemente de sua localização na imagem). No entanto em aplicações envolvendo PLN esta identificação ainda não é tão fácil de ser realizada, uma vez que em muitos idiomas, partes das frases podem ser separadas por várias outras palavras e a composição de uma sentença pode ser feita de diferentes maneiras, um adjetivo, por exemplo, pode modificar um determinado substantivo. Logo, cria-se então uma oportunidade de se obter este tipo de conhecimento através da rede CNN.

A outra rede, ELM, proposta neste trabalho visa usar o conhecimento obtido através da CNN (mapas de características) para classificar dados textuais financeiros obtidos através de portais de notícias brasileiros e assim tentar entender e predizer o comportamento futuro do mercado financeiro brasileiro.

Unido ao entendimento da CNN e ELM é possível citar também a importância do sistema estudado neste trabalho. O mercado financeiro, em especial o mercado de ações brasileiro, além de ser um sistema altamente complexo, dinâmico e não linear é um dos mercados de maior liquidez dentre os mercados emergentes e muito pouco explorado ainda, devido principalmente à dificuldade de se obter dados confiáveis para o entendimento do mesmo.

Cabe salientar também que, até o presente momento, é a primeira vez que a metodologia proposta aqui, será utilizada em processamento de textos financeiros escritos na língua portuguesa e para prever o mercado de ações brasileiro.

1.2 Estrutura da tese

A estrutura do trabalho proposto está organizada da seguinte forma: os capítulos 2, 3, 4 e 5 estão relacionados a uma revisão teórica que abrange todos os conceitos necessários para o entendimento da metodologia proposta, assim como também são apresentados os trabalhos correlatos. Já os capítulos 6,7 e 8 estão relacionados aos desenvolvimentos realizados e resultados obtidos. Mais especificamente pode-se dizer que:

- O capítulo 2 aborda alguns conceitos e informações importantes sobre o mercado de ações, onde ênfase é dada ao mercado de ações brasileiro.
- Nos capítulos 3 e 4 estão apresentados todos os métodos e técnicas que apoiam aplicações envolvendo o uso de mineração de textos para a previsão de mercados financeiros de uma maneira geral. Nestes capítulos ênfase foram dadas às técnicas de RNAs e métodos de representação das palavras.
- O capítulo 5 apresenta alguns trabalhos correlatos relacionados à previsão de mercados financeiros mundiais utilizando as técnicas citadas nos capítulos anteriores (3 e 4). Foram evidenciados apenas trabalhos relevantes e recentes que de alguma forma estão envolvidos com as técnicas utilizadas para atingir os objetivos deste trabalho.
- Nos capítulos 6 e 7 estão apresentados toda a metodologia utilizada para o estudo em questão, assim como os experimentos realizados, os resultados obtidos e possíveis discussões.
- Por fim o capítulo 8 conclui a tese e apresenta algumas propostas de trabalhos futuros

2. Mercado de Ações no Brasil

A bolsa de valores pode ser definida como um clube de corretores de valores, intermediários e outros agentes que intermediam a compra e venda de ativos financeiros (MAURO, 2001). Os negócios e transações realizadas na bolsa de valores são públicos e podem ser acompanhados pelos meios de difusão em massa. A bolsa brasileira conhecida como BM&FBOVESPA nasceu em 2008 a partir da integração das operações da Bolsa de Valores de São Paulo e da Bolsa de Mercadorias e Futuros. Mais recentemente a bolsa brasileira se modernizou com o objetivo de potencializar toda uma variedade de oportunidades de negócios em um ambiente moderno de integração global. Como consequência nasceu a B3 ou bolsa brasileira como uma das mais importantes bolsas da América Latina. A B3 nasceu fruto da combinação entre a BM&FBOVESPA e a Cetip e tem como objetivo principal um compromisso com a inovação e com o constante desenvolvimento do mercado financeiro e de capitais (BMF, 2016). Em essência a B3 (Brasil, Bolsa e Balcão) é formada por um grupo de empresas dentre as quais podemos citar o banco BM&FBOVESPA (subsidiaria integral da Bolsa de Mercadorias & Futuros); BM&F (USA) Inc. (representante da BM&FBOVESPA no exterior); BM&FBOVESPA(UK) Ltda; BVRJ a qual representa uma bolsa inativa; uma organização da sociedade civil de interesse público ou Instituto BM&FBOVESPA e por fim a BM&FBOVESPA Supervisão de Mercados (BSM), (BMF2, 2016).

Segundo dados operacionais a bolsa B3 pode ser considerada a quinta maior bolsa de mercado de capitais e financeiro do mundo em valor de mercado, com patrimônio de aproximadamente US$ 13 bilhões. Aproximadamente entre sete e oito bilhões de reais são movimentados todos os dias na bolsa de valores. Na figura 2.1 é possível visualizar a média de volume financeiro negociada na bolsa entre julho de 2016 e julho de 2017.

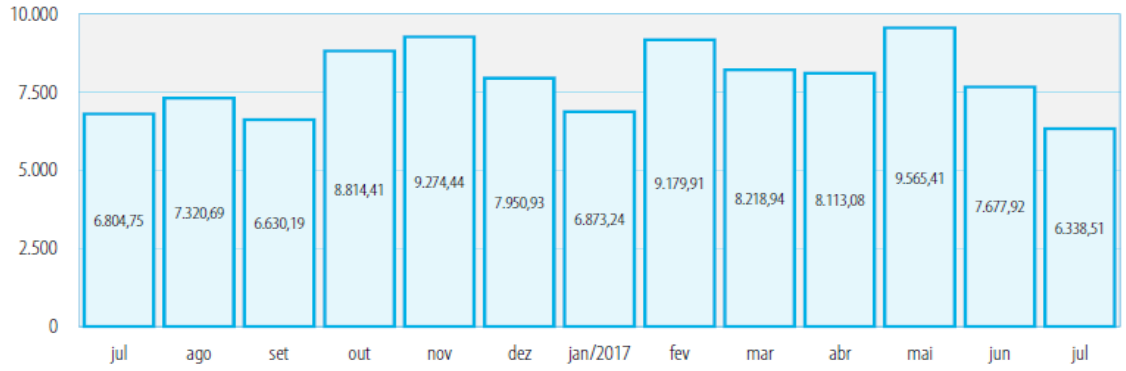

Figura 2.1 - Médias diárias do volume financeiro em um ano quantificadas em milhões de reais (BMF3, 2016)

Nos últimos anos, de maneira geral, o crescimento físico de investidores (“pessoas físicas”) tem aumentado consideravelmente. Inclusive, em anos recentes, apesar da crise econômica no país este número ainda é significativo se comparado aos padrões de anos anteriores. Como um exemplo o volume financeiro negociado por pessoa física entre os dias 03 e 25 de julho de 2017 em operações (compra + venda) foi de aproximadamente 215 milhões de reais.

Na Bolsa de Valores são negociadas ações de companhias brasileiras de capital aberto. A ação representa um valor mobiliário emitido por sociedades anônimas (empresa) na qual pode ser considerada como a menor fração do Capital Social da empresa. A empresa emite ações com o objetivo de captar recursos para desenvolver projetos que levem ao desenvolvimento da empresa. O investidor quando adquire uma ação de uma companhia na Bolsa de Valores, ele torna-se sócio da companhia, mesmo que em uma proporção bem pequena em relação ao controlador (sócio majoritário). Em geral no mercado acionário brasileiro existem dois tipos de ações, as chamadas ordinárias (ON) e as preferenciais (PF). Por exemplo, o acionista detentor de uma ação preferencial tem preferência no recebimento de dividendos (percentual sobre o lucro da companhia), porém não tem direito ao voto nas decisões do Conselho Administrativo. Já as ações ordinárias (ON) concedem àqueles que são acionistas o direito a voto nas assembléias de acionistas (decisões da empresa), mas não têm preferência na distribuição de resultados.

2.1. Mercado de Ações – Características e Séries Temporais

O preço de uma ação é basicamente determinado pela lei da oferta e da procura com relação às ações da empresa. O preço de uma ação para um instante de tempo representa o valor da transação entre um comprador do título e um vendedor. Desde que exista procura no mercado acionário, as ações podem ser convertidas em dinheiro a qualquer momento por intermédio de uma Sociedade Corretora (comumente chamada de corretora) e através da negociação na Bolsa de Valores. As ações de empresas que são muito negociadas são chamadas de *blue chips*, enquanto aquelas pouco negociadas são conhecidas como *small caps*. A BM&FBOVESPA possui segmentos específicos para listar as companhias, por exemplo, Bovespa Mais Nível 2, Novo Mercado, etc. Os diferentes segmentos de negociação estabelecem regras rígidas de governança corporativa. As ações são identificadas por quatro letras maiúsculas seguidas de um número, por exemplo, PETR4. As letras representam ou identificam a empresa enquanto os números representam o tipo de ação, usualmente ordinária ou preferencial. Por exemplo, para a Petrobras existem os códigos PETR3 e a PETR4, onde a primeira identifica ações ordinárias de emissão da companhia, enquanto o segundo código representa as ações preferenciais.

Para avaliar o sentimento do mercado acionário como um todo é necessário um indicador que reflita o comportamento geral do mercado. No mercado de ações brasileiro este indicador (ou índice) é conhecido como Ibovespa e representa o desempenho médio dos preços das ações mais negociadas na bolsa. Cada ação contribui com um fator, o qual é ponderado proporcionalmente ao volume negociado das ações. E em geral as ações negociadas no mercado acompanham a evolução deste indicador (Ibovespa), mesmo que muitas delas não façam parte do mesmo. Vale ressaltar que na maioria das vezes os indicadores (como por exemplo, o Ibovespa) são os mais estudados em análises referentes ao comportamento do mercado acionário.

As oscilações do Ibovespa (juntamente com o preço das ações das empresas negociadas na B3) oscila durante um dia de negociação em função de muitos fatores, como notícias econômicas, eventos políticos e econômicos, além de muitos outros fatores de diferente natureza. Uma notícia repentina pode levar subitamente a uma grande procura por ações de uma empresa, o que por sua vez leva ao aumento dos preços das mesmas. Se esta

ação forma parte do Ibovespa, este aumento sólido do preço da ação da empresa pode contribuir para a valorização do Ibovespa, ou até mesmo, para uma reação em cadeia que também leve à valorização do indicador.

A cada dia de negócio no mercado acionário o preço das ações oscila desde a abertura (preço de abertura) passando por um mínimo e máximo, até o fechamento do mercado (preço de fechamento). Este comportamento, ou seja, estes preços característicos podem ser representados por um desenho de uma estrutura chamada de *candlestick*. Esta estrutura e suas representações (máximo, mínimo, fechamento e abertura) é apresentada na figura 2.2 e de maneira geral a mesma representa graficamente a variação dos preços de uma determinada ação em um determinado período de tempo (diário, semanal, etc.). No caso do Ibovespa o *candlestick* reflete os valores do indicador ao longo do período de tempo. Uma questão interessante é que se o preço de fechamento é maior que o preço de abertura o *candlestick* é azul, caso contrário vermelho.

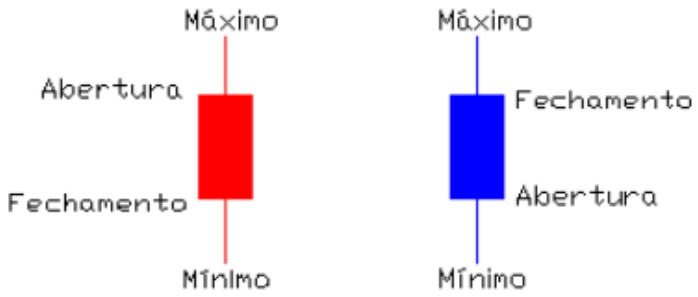

Figura 2.2 - *Candlestick* e suas representações

Como o mercado funciona com uma periodicidade diária, os valores dos títulos negociados evoluem diariamente conforme a situação do mercado acionário. Isto leva a uma sequência de estrutura *candlesticks*, o qual forma uma série temporal da mesma. A figura 2.3 representa um gráfico de *candlesticks* correspondente à série temporal dos valores diários do Ibovespa ao longo do ano 2017. Note que nesta figura os *candlesticks* de alta são representados em verde, enquanto os de baixa são vermelhos.

Figura 2.3 - Gráfico de *candlestick*: Ibovespa (Período de março a agosto de 2017)

É importante salientar a importância do Ibovespa, dado que o mesmo reflete de uma maneira geral o comportamento do mercado de ações como um todo. A primeira informação sobre qualquer estudo referente ao mercado acionário vem da observação direta da série temporal que representa o mesmo (ou seja, o Ibovespa). Uma apreciação permite identificar, por exemplo, a tendência do indicador. Este fato pode ajudar a prever a tendência das principais ações do mercado, ou "*blue chips*". Por este motivo salienta-se mais uma vez que este indicador é tido como objeto de estudo na maioria dos trabalhos científicos sobre o mercado acionário brasileiro.

2.2. Estudo do Mercado Acionário

O investimento em ações pode ser considerado um investimento de alto risco dado o caráter aleatório do movimento do preço das ações. A liquidez de cada ação e seu preço estão intimamente ligados ao momento presente da companhia e ao cenário interno e /ou global. As aquisições, vendas e/ou fusões de companhias são os principais estímulos aos preços das ações. Outros fatores de diferente natureza, como fenômenos climáticos ou geopolíticos, por exemplo, podem também alterar substancialmente o preço das ações.

Como consequência, descobrir como o preço das ações irá se modificar não é uma tarefa trivial. Neste sentido, diferentes estudos e teorias têm sido desenvolvidos para tentar explicar o comportamento dos mercados acionários, auxiliando assim os investidores na sua tomada de decisão. Deve-se destacar que estas teorias podem também serem aplicadas a outros sistemas estocásticos cujos comportamentos podem também serem identificados por

meio de séries temporais de comportamento altamente não linear, como por exemplo, eletrocardiogramas.

Por último, dentre os modelos ou hipóteses relevantes para o estudo de mercados financeiros é possível citar a hipótese do *Mercado Eficiente*. Esta teoria está no centro das discussões, onde a principal analise está relacionada a possibilidade de que o comportamento dos mercados acionários pode ou não ser previsto.

2.2.1. Hipótese do mercado Eficiente

A hipótese do mercado eficiente (EMH) foi introduzida por (FAMA, 1964). Em essência ela postula que em um mercado acionário eficiente o preço atual dos ativos reproduz toda a informação disponível, ou seja, as cotações dos ativos refletem toda a informação conhecida sobre a empresa ou o mercado de forma geral. Sendo assim o passado não contém qualquer informação que já não esteja incorporada no preço atual dos ativos. FAMA (1964) propôs três formas de eficiência de mercado (fraca, semiforte e forte). A forma fraca mostra que somente os preços passados e informação histórica estão incorporados no preço corrente dos ativos. Na forma semiforte os preços das ações (ou outros ativos) incorporam o comportamento passado dos preços, assim como também incorporam o restante da informação publicada sobre os ativos. Estas informações podem ser notícias específicas ou anúncios de outra natureza, como por exemplo, distribuição de lucros sobre a forma de dividendos. Por fim, a forma forte, na qual os preços refletem não só a informação pública, mas toda a informação que pode ser obtida, inclusive as informações consideradas ocultas ou privilegiadas.

2.2.2. Análise Fundamentalista e Técnica

Analises de diferentes pontos de vista podem ser úteis na previsão do comportamento geral do preço de ativos de uma companhia. De modo geral, a ideia é prever a tendência geral do preço de um determinado ativo. Neste sentido, a análise fundamentalista enfatiza principalmente os indicadores de desempenho financeiro de uma companhia reportados em seus balanços financeiros dando uma ideia sobre a saúde financeira da empresa.

A idéia destes estudos é que a partir do conhecimento dos indicadores e a sua evolução temporal é possível estipular uma correlação com o comportamento temporal do preço das ações, ou mesmo, estabelecer modelos de previsão os quais tentam incorporar estimativas financeiras em variáveis fundamentalistas fornecendo previsões indiretas sobre o comportamento acionário. Em outras palavras, o objetivo é reunir e interpretar estas informações e agir antes que a informação seja incorporada no preço das ações. Do ponto de vista de investimento, este intervalo de tempo entre um evento e a resposta de mercado apresenta uma oportunidade de negociação.

Na análise fundamentalista também são levadas em consideração informações gerais, dados macroeconômicos do país, planos e as estimativas da empresa, além de muitos outros. Neste sentido é possível afirmar que as notícias financeiras têm uma importância muito grande para investidores que utilizam análise fundamentalista, pois as mesmas descrevem os fatores que podem afetar o preço do ativo. Como um exemplo de indicador fundamentalista pode se mencionar o indicador P/L = preço por ação/lucro anual por ação, que representa uma estimativa do tempo em anos que o investimento em uma ação retornaria ao investidor sobre a forma de dividendos, considerando que ambos os parâmetros sejam estáveis ao longo do período temporal.

A análise técnica por sua vez é baseada em dados estatísticos de séries temporais numéricas referentes aos preços das ações e outros parâmetros de negociação como volume, número de negócios, etc. Dentre estes indicadores é possível citar o índice de força relativa (*IFR*) que está relacionado às variações do preço das ações, médias móveis (exponenciais ou não) do preço do ativo analisado, *OnBalance Volume*, entre outros.

A análise técnica utiliza estes indicadores para a tomada de decisão na hora de investir, ou seja, determinar tendências, pontos de entrada ou saída no mercado. Nesta análise padrões de comportamento como triângulos de acumulação, retângulos, padrões ombro-cabeça-ombro, entre outros, ajudam a determinar o comportamento futuro da bolsa. A principal estratégia na análise técnica está relacionada à identificação destes padrões em séries temporais. Deve-se destacar que segundo a análise técnica, as notícias divulgadas no dia a dia não influenciam na previsão.

Com base nesta abordagem, e levando em consideração o histórico dos preços dos ativos estudados, diferentes métodos de previsão podem ser implementados, como por exemplo, no trabalho de (FARIA *et al.*, 2009). Neste sentido, outras ferramentas como RNAs, por exemplo, podem ser implementadas a fim de usar o preço das ações para prever seu comportamento futuro.

3. Redes Neurais Artificiais

A criação de computadores inteligentes tem sido o foco de pesquisas de mais da metade do século XX. Desde então, vários esforços têm sido realizados no sentido de simular a forma como o cérebro humano pensa e implementá-la em computadores através da Inteligência Artificial (IA). Dentro do campo da IA uma das técnicas que vem se destacando muito, em especial nos últimos anos, são as RNAs.

As RNAs são modelos matemáticos que de alguma forma tentam imitar as estruturas neurais biológicas e possuem capacidade computacional adquirida através do aprendizado e generalização (HAYKIN, 2001). Pode-se dizer então que as RNAs são capazes de reconhecer e classificar padrões e posteriormente generalizar o conhecimento adquirido.

Os primeiros modelos de RNAs (as redes MLPs), também conhecidos atualmente como modelos de arquiteturas rasas (conforme citado anteriormente) só foram amplamente compreendidos em meados da década de 80. Desde então, os mesmos foram utilizados em diferentes aplicações envolvendo principalmente a tarefa de reconhecimento de padrões em diferentes tipos de dados como imagens, áudio e texto.

Entretanto, nos últimos anos surgiu uma nova abordagem de rede neural baseada em modelos de aprendizado profundo. Esta nova abordagem de Redes neurais (conforme citado anteriormente - DLNN), exploram a propriedade de que uma boa representação interna deve ser hierárquica, onde, as características de mais alto nível podem ser obtidas a partir da composição de níveis mais baixos. Este conceito hierárquico com múltiplos estágios de abstração está em linha com estudos sobre o cérebro humano, em especial ao sistema visual (GOODFELLOW *et al*., 2016). Os resultados obtidos com estes novos modelos vêm apresentando melhorias impressionantes principalmente em aplicações de reconhecimento de imagem (KRIZHEVSKY *et al*., 2012) e áudio (GRAVES *et al*., 2013), assim como alguns resultados promissores na área de PLN (RIE *et al*., 2014).

3.1. Histórico – De arquiteturas rasas a arquiteturas profundas

As RNAs foram inicialmente desenvolvidas por McCulloch e Pitts em 1943, quando eles formalmente descreveram pela primeira vez como seria um neurônio artificial (MCCULLOCH *et al*., 1943). Um neurônio é usado para representar uma unidade computacional que recebe alguns valores de entrada (representados pelos dendritos no modelo biológico) e produz uma saída (representando o axônio no modelo biológico), baseada em uma função de ativação conforme ilustrado na figura 3.1.

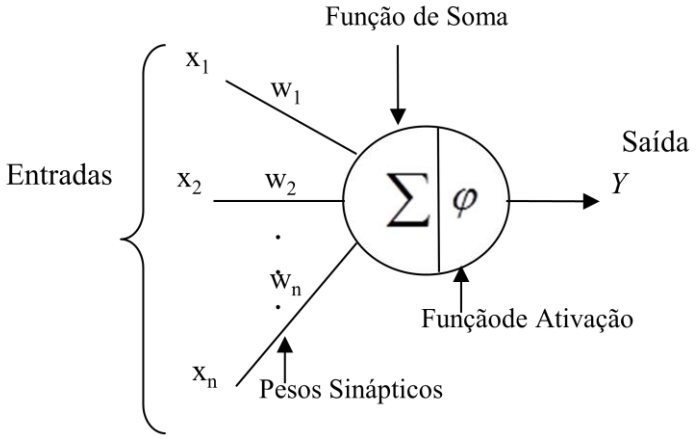

Figura 3.1 - Modelo de um Neurônio Artificial (MCP) proposto por (MCCULLOCH *et al*., 1943)

Conforme pode ser visualizado na figura 3.1, o modelo é formado por um vetor de entradas ($x_1, x_2,... x_n$) que estão associados a seus respectivos pesos sinápticos ($w_1, w_2,... w_n$) cujos valores podem estar em um intervalo que inclui tanto valores positivos como negativos, sendo que cada entrada pode ser então inibitória ou excitatória. Um neurônio dispara quando a soma dos valores do vetor de entrada, ponderados pelas respectivas sinapses do neurônio ultrapassa o seu limiar de excitação (*threshold*), ou seja, a ativação do neurônio é dada pela soma ponderada das entradas submetida a uma função de ativação que determina se a soma é maior ou menor que o limiar, o que pode ativar ou não, a saída do neurônio. O neurônio MCP obedece à lei do "tudo ou nada", sempre estará em um de dois estados: ativado ou desativado (1 ou 0) e as entradas são recebidas de forma síncrona (HAYKIN, 2001). A função de ativação do modelo MCP é dada pela função de limiar descrita na equação 3.1

$$\sum_{i=1}^{n} x_i w_i \geq \theta \tag{3.1}$$

Onde *n* é o número de entradas do neurônio, w_i *é* o peso associado à entrada x_i, e θ é o limiar (*threshold*) do neurônio.

Diferentemente do modelo do neurônio biológico, onde não existe sincronismo e nem ativação em tempo discreto, MCCULLOCH *et al*. (1943) simplificaram seu modelo assumindo que os neurônios em cada camada da rede disparam sincronicamente e que as entradas em instante *t* produzem a sua saída no tempo *t* + 1. Outro conceito fundamental que também faz parte do neurônio biológico e não apresentado no MCP foi a ausência de uma técnica de aprendizado, sendo assim com valores de pesos sinápticos fixos (não ajustáveis) e apenas uma camada, o MCP só conseguia implementar funções linearmente separáveis.

No final da década de 40, Donald Hebb melhorou o MCP propondo a primeira regra de aprendizagem para as RNAs (HEBB, 1949). Dentro do contexto neurobiológico, Hebb (1949) sugeriu que a conectividade do cérebro é continuamente modificada de acordo que o organismo vai aprendendo tarefas funcionais diferentes e que agrupamentos neurais são criados por estas modificações (HAYKIN, 2001). Sendo assim, o postulado de aprendizagem de (HEBB, 1949) afirma que se os dois neurônios, participantes de uma sinapse têm ativação simultânea, então a força da conexão entre eles deve ser seletivamente aumentada ou enfraquecida e eliminada quando ocorre uma ativação assíncrona.

A forma mais simples da regra de aprendizado de Hebb é descrita por (HAYKIN, 2001) como:

$$\Delta w_{ki}(n) = \eta y_k(n) x_i(n) \tag{3.2}$$

Onde η é uma constante positiva que determina a taxa de aprendizado, x_i representa o sinal pré-sináptico, y_k representa o sinal pós-sináptico do neurônio *k*, *n* a iteração do algoritmo de aprendizado e w_{ki} representa o peso sináptico que faz a conexão entre o sinal de entrada x_i e o neurônio *k*.

ROSENBLATT (1958) introduziu o *perceptron*, uma rede neural com neurônios MCP dispostos em camadas e implementado para resolver problemas de reconhecimento de padrões. O modelo consiste de três camadas, onde a primeira camada recebe as entradas do exterior e possui conexões fixas, a segunda camada recebe os impulsos da primeira camada através de conexões com pesos ajustáveis e por fim a terceira camada ou camada de saída. O objetivo de um *perceptron* é aprender através de seus pesos sinápticos (parâmetros livres do modelo) de tal forma que a unidade de saída consiga produzir a saída correta para cada

entrada do modelo através de atualizações iterativas de seu algoritmo de treinamento conhecido como algoritmo de convergência do *perceptron* (ROSENBLATT, 1958). É possível considerar então que a grande contribuição deste modelo foi a introdução deste algoritmo de treinamento que possibilitou a classificação de padrões linearmente separáveis, isto é, padrões que estão situados em lados opostos de um hiperplano.

Em 1965 houve a primeira tentativa de se simular uma rede de múltiplas camadas de características não lineares (DLNN) com o trabalho de (IVAKHNENKO, 1968, 1971; IVAKHNENKO *et al*., 1966; IVAKHNENKO *et al*., 1967). As redes desenvolvidas pelos autores foram denominadas como *Group Method of Data Handling* (GMDH) ou Redes Neurais Polinomiais. Segundo (LIU *et al*., 2000) as redes GMDH são capazes de manipular dados nos quais sistemas complexos podem ser decompostos em unidades menores e mais simples, sendo agrupados em várias camadas. As entradas da rede são manipuladas através de combinações, para em seguida serem enviadas aos nós da rede. Esses nós, são responsáveis por avaliar um número limitado de entradas através de uma função polinomial e gerar uma saída que vai ser encaminhada como entrada aos nós subsequentes da próxima camada.

Além das redes GMDH, outro modelo de RNA que ganhou destaque nos anos 70 e incorpora bem os conceitos de uma rede biologicamente inspirada e merece o atributo de profundo (*deep*) são as redes convolutivas (atualmente denominadas como Redes Neurais Convolucionais - CNNs). Estas redes foram primeiramente utilizadas por (FUKUSHIMA, 1975) e conhecidas como *Neocognitron*. As mesmas fizeram uso de um projeto hierárquico de várias camadas que permitiu ao computador "aprender" padrões visuais. Estas redes são muito similares a versões modernas (CNNs), pois, possuem várias camadas de convolução e camadas de *subsampling* (ou *downsampling*), entretanto, as mesmas diferentemente das versões atuais (que na maioria das vezes fazem uso do algoritmo *backpropagation*) foram treinadas por regras de aprendizado não supervisionadas locais (FUKUSHIMA, 2013). Além disso, o projeto de Fukushima permitia atribuir características importantes de cada imagem manualmente aumentando assim os pesos de certas conexões.

Até o final dos anos 60 e ínicio do anos 70 muito ainda se falava sobre o impacto que teve o perceptron na área de RNAs. Entretanto em 1969 MINSK *et al*. (1969) divulgaram um livro onde o modelo foi muito criticado. Neste livro denominado "*Perceptrons*", os autores fizeram uso da matemática para demonstrar que o perceptron não poderia ser treinado para reconhecer muitas classes de padrões, pois essa rede de camada única era capaz de aprender apenas padrões linearmente separáveis. Isto adicionado ao fato de que as funções XOR ("ou-

exclusivo") e XNOR não puderam ser representadas, as décadas de 1970 e 1980 sofreram um declínio no uso das redes neurais para o aprendizado de máquinas (HAYKIN, 2001).

Os avanços das pesquisas em RNAs ressurgiram apenas em meados dos anos 80 com algoritmos de aprendizado mais eficientes para redes multicamadas, como a descrição do modelo de (HOPFIELD, 1982; 1984), as Máquinas de Boltzmann (ACKLEY *et al*., 1985) e o surgimento do algoritmo *backpropagation* (RUMELHART *et al*., 1986), sendo este último o grande responsável por popularizar o treinamento dos *perceptrons* de múltiplas camadas e consequentemente o avanço do uso das RNAs em diferentes áreas de atuação como na compreensão de imagens e voz, no reconhecimento de caracteres manuscritos, no diagnósticos médicos e no processamento de sinais e análise de séries temporais.

No entanto, todos estes avanços ainda estavam relacionados a utilização apenas de RNAs com uma quantidade reduzida de camadas de processamento (MLP com uma ou duas camadas escondidas) dificultando assim o seu poder de modelar e armazenar informações principalmente quando se tratava de grande quantidade de dados e também de dados com complexos comportamentos não lineares.

No começo dos anos 90, LECUN *et al*. (1989) fizeram uma tentativa de se implementar uma rede neural com várias camadas fornecendo uma demonstração prática do uso das RNAs convolutivas, as mesmas propostas por (FUKUSHIMA, 1975), porém juntamente com o algoritmo de treinamento *backpropagation*. A ideia foi utilizar estas redes para a tarefa de reconhecimento de padrões, mais precisamente no reconhecimento de CEPs manuscritos. Os autores trabalharam com uma grande quantidade de dados e apesar das limitações de hardware que existia na época (foram necessários aproximadamente 3 dias para treinar a rede) foi possível demonstrar que a rede neural implementada foi inteiramente capaz de executar a tarefa. E mais importante ainda, foi que esta combinação de técnicas deu um passo importante no surgimento das redes CNNs atuais.

Outros avanços significativos também ocorreram nos anos 90 como o desenvolvimento das máquinas de Vetor de Suporte (SVM) um sistema para mapear e reconhecer dados semelhantes (CORTES *et al*., 1995) e o desenvolvimento da *Long Short-Term Memory* (LSMT), um outro tipo de arquitetura de redes neurais conhecidas como Redes Neurais Recorrentes (HOCHREITER *et al*., 1997).

LECUN *et al*. (1998) publicaram um trabalho onde foi realizada uma revisão de vários dos métodos que foram utilizados no trabalho divulgado en 1989 (LECUN *et al*., 1989) para reconhecimento de caracteres manuscritos. Neste trabalho, os autores

apresentaram um sistema de reconhecimento composto de múltiplos módulos (extração de características, segmentação, reconhecimento e modelagem da linguagem) e um novo paradigma de aprendizado denominado *Graph Transformer Networks* (GTN) que permitia que estes módulos fossem treinados globalmente usando métodos baseados em gradientes de modo a minimizar uma medida de desempenho geral. Vários testes foram realizados usando as redes CNNs proposta no trabalho anterior juntamente com a GTN e diferentes *benchmarks* de dígitos manuscritos. Os resultados obtidos foram considerados como robustos e eficientes. Apesar destas tentativas de se simular redes neurais com um número maior de camadas de processamento e da obtenção de resultados considerados como robustos, assim como no trabalho anterior, a técnica ficou limitada devido a disponibilidade de recursos de computação disponíveis na época.

O próximo passo significativo para o aprendizado profundo ocorreu em 1999, com o desenvolvimento das Unidades de Processamento Gráfico (GPUs - *Graphic Processing Units*) que permitiu aumentar a velocidade no processamento de dados. Isto fez com que muitos pesquisadores investissem no treinamento de RNAs com múltiplas camadas (redes profundas) utilizando o algoritmo *backpropagation* para o treinamento das redes e tangente hiperbólica e função sigmóide como função de ativação. Inicialmente estas pesquisas obtiveram pouco sucesso devido a um problema conhecido como problema de dissipação do gradiente (*vanishing gradient problem*) o que dificulta a convergência da rede. Este problema, de alguma forma, pode ser influenciado pela escolha das funções de ativação utilizadas no modelo. As funções de ativação comumente utilizadas nos modelos das RNAs (tangente hiperbólica e sigmoide) mapeiam suas entradas para um intervalo muito pequeno (por exemplo: [-1,1] no caso da tangente hiperbólica e [0,1] no caso da sigmoide). Isto resulta em grandes regiões no espaço de entrada sendo mapeados para um intervalo extremamente pequeno. Sendo assim, mesmo uma grande mudança nesta região no espaço de entrada produzirá apenas uma pequena alteração na saída (gradientes pequenos). Isto se agrava ao longo de uma rede com muitas camadas, ou seja, uma mudança de valores na entrada, pouco vai alterar as camadas mais distantes.

Em 2006, pesquisadores do *Canadian Intitute for Advanced Research* (CIFAR) publicaram um artigo que foi visto como um avanço significativo para a era *deep learning*, marcando assim o início desta área tão importante nos dias atuais.

Neste artigo, os autores introduziram uma rede denominada *Deep Belief Networks* – DBNs (HINTON *et al*., 2006) com um algoritmo de aprendizado rápido que treina uma

camada escondida por vez usando a abordagem do aprendizado não supervisionado para cada camada, uma Máquina de Boltzmann Restrita (*Restricted Boltzmann Machine* - RBM), (FREUND *et al*., 1992). As RBMs podem ser treinadas de forma eficiente por um algoritmo introduzido por (HINTON, 2002). De uma forma geral, as RBMs são treinadas para maximizar o produto das probabilidades atribuídas a um determinado conjunto de treinamento.

No ano seguinte, ainda dentro da mesma abordagem introduzida por (HINTON *et al*., 2006), ou seja, orientar o treinamento de níveis intermediários de representação usando a aprendizagem não supervisionada, foram propostos os algoritmos baseados em *auto-encoders* (POULTNEY *et al*., 2007; BENGIO *et al*., 2007). Nestes trabalhos, os autores apresentaram um forte argumento de que métodos de aprendizado profundo, ou seja, métodos com muitas camadas de processamento ou métodos com representações de características hierárquicas dos dados são mais eficientes para lidar com problemas difíceis e complexos do que os métodos considerados como modelos de arquitetura rasa (RNAs MLP, SVM, etc).

Dentro deste contexto é possível afirmar então que a partir desta nova abordagem proposta por (HINTON *et al*., 2006); com o aumento da velocidade das GPUs (que fez com que o processamento paralelo fosse mais rápido, mais barato e mais poderoso) e a grande quantidade de dados disponíveis nos dias atuais (*Big Data*) seja na forma de imagens, textos, mapas, entre outros, fez com que a área de *deep learning* e consequentemente a Inteligência Artificial aumentasse sua popularidade permitindo assim o uso das mesmas em diferentes aplicações.

De fato, BENGIO, (2009) afirma que desde de 2006 as redes neurais *deep learning* vem sendo aplicada com sucesso em diferentes áreas de atuação, como na tarefa de classificação (AHMED *et al*., 2008; LAROCHELLE *et al*., 2007; BOUREAU *et al*., 2008), na tarefa de regressão (HINTON *et al*., 2008), na redução da dimensionalidade dos modelos (HINTON, 2006), no processamento de linguagem natural (COLLOBERT *et al*., 2008), entre outras. E que embora as redes *auto-encoders*, RBMs e DBNs possam ser treinadas com dados não classificados, em muitas aplicações citadas anteriormente, as mesmas também estão sendo utilizadas com sucesso na inicialização de redes neurais MLPs aplicada a uma determinada tarefa.

Cabe destacar que apesar de toda esta popularidade, um pouco antes destas pesquisas chegarem na indústria (entre 2009 e 2010), algumas perguntas ainda faziam parte do cotidiano dos pesquisadores, como por exemplo, porque o algoritmo *backpropagation* ainda

não trabalhava tão bem como em modelos de arquiteturas rasas? Tentando entender este problema, GLOROT *et al*. (2010) publicaram um artigo onde foi destacado a importância do tipo de função de ativação utilizada nos modelos de redes neurais. O que culminou em uma outra dúvida, qual seria então a função de ativação ideal para ser utilizada nos modelos de rede neurais? Respondendo as estas perguntas alguns autores (JARRETT *et al*., 2009; GLOROT *et al*., 2011; NAIR *et al*., 2010) conseguiram chegar a uma mesma conclusão, isto é, que a melhor função de ativação a ser utilizada em modelos de redes neurais é a função denominada função Unidade Retificadora Linear (*Rectified Linear Unit* – ReLu), (equação 3.8), uma função extremamente simples mas que acelara muito o treinamento das redes neurais.

Os autores citados anteriormente fizeram vários experimentos utilizando a função ReLu e demonstraram que o erro de treinamento gerado pelas redes que utilizam este tipo de função de ativação é menor, devido ao fato da mesma não ser tão suscetível ao problema de dissipação dos gradientes (o que ocorre com as funções ativação tangente hiperbólica e sigmóide). Os autores destacaram também que o cálculo computacional da função é fácil e rápido, qualquer valor negativo é definido como zero sem envolver exponenciação ou mesmo qualquer operação de multiplicação e ou divisão. Isto faz com que a convergência dos parâmetros seja muito mais rápida. Outro ponto importante a ser destacado também que o uso da função ReLu nos experimentos realizados permitiu o treinamento supervisionado de redes neurais *deep* (como por exemplo, a rede MLP com mais de duas camadas escondidas) sem a necessidade de se utilizar o pré-treinamento não supervisionado.

Dentro deste contexto que envolve desde a descrição formal do neurônio biológico proposto por (MCCULLOCH *et al*., 1943) até os dias atuais é possível afirmar que a área de IA, em especial a área relacionada a RNA está em crescente expansão. As redes denominadas MLP e as redes CNN tem grande contribuição neste progresso. Sendo assim, nas próximas seções as mesmas serão detalhadas, em especial a rede CNN que faz parte da metodologia desta tese, assim como também será feita uma descrição da Rede ELM que também está no escopo deste trabalho.

3.2. Redes Neurais de Múltiplas Camadas

As redes neurais de múltiplas camadas (MLPs) podem ser consideradas como as mais importantes arquiteturas da classe de redes neurais. Elas constituem uma generalização do

perceptron de camada única proposto por (ROSENBLATT, 1958) e citado na seção anterior desta tese. Consiste de uma camada de entrada, uma ou mais camadas de neurônios escondidos e uma camada de saída. Um exemplo da arquitetura de uma rede MLP totalmente conectada (onde um neurônio em qualquer camada da rede está conectado a todos os neurônios da camada anterior) com uma única camada escondida e uma camada de saída pode ser visualizada na figura 3.2.

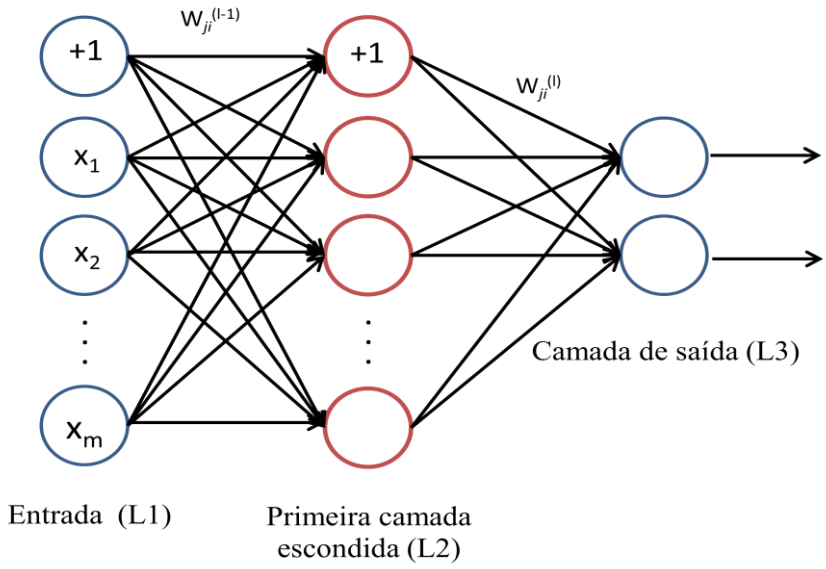

Figura 3.2 - Arquitetura de uma rede MLP com uma camada escondida

A figura 3.2 apresentada é composta por *m* padrões ($x_1, x_2, \ldots x_m$) na camada de entrada (L1), uma camada escondida L2 com *n* neurônios (que recebe estímulos do ambiente – L1) e uma camada de saída L3 (que recebe estímulos da camada anterior - L2) com dois neurônios representando as saídas da rede. Os valores $w_{ji}^{(l)}$ representam os pesos sinápticos associados a conexão entre o neurônio *i* na camada *l*-1 e o neurônio *j* na camada *l*.

O treinamento da rede MLP é realizado de forma supervisionada através do algoritmo *backpropagation* proposto por (RUMELHART *et al*., 1986). Este algoritmo é baseado na regra de aprendizagem por correção de erro e consiste de duas fases: uma fase de propagação adiante (*Forward*) e uma fase de propagação reversa (*Backward*) conforme pode ser visualizado na figura 3.3. Na fase *Forward*, os sinais de uma determinada amostra ($x_1, x_2, \ldots x_m$) são inseridos nas entradas da rede. Os mesmos são então propagados através da rede, camada por camada até a produção de suas respectivas saídas (resposta da rede). Os valores atuais de pesos sinápticos e limiares considerados nesta etapa são todos fixos e permanecem inalterados durante a execução desta fase. As respostas produzidas pelas saídas da rede são comparadas com as respectivas respostas desejadas e um sinal de erro é produzido. Baseados

nos sinais de erros encontrados entre as respostas desejadas e aquelas obtidas pelos neurônios de saída, na fase *backward* os pesos e limiares dos neurônios são ajustados utilizando uma regra de correção de erro. Mais detalhes sobre o treinamento da MLP utilizando o algoritmo *backpropagation* poderá ser visualizado nas próximas seções.

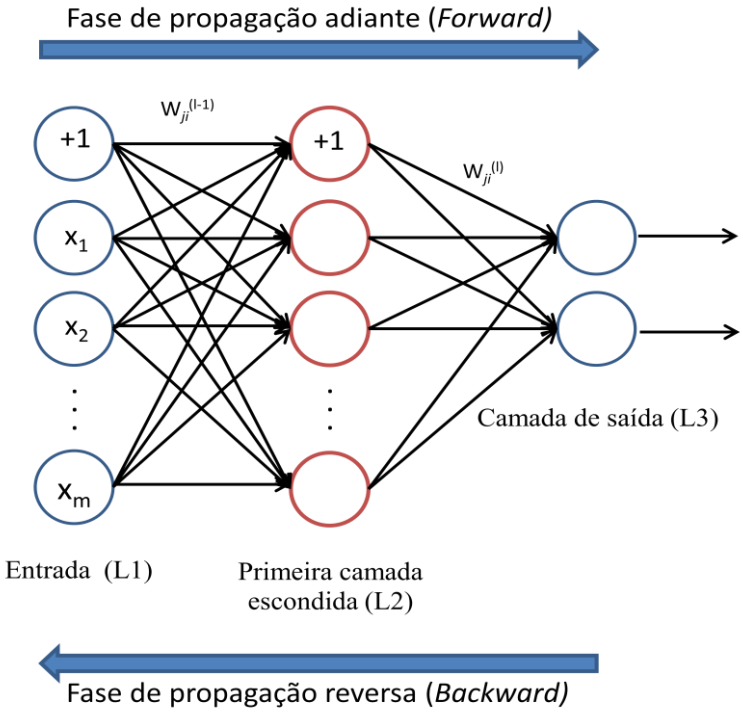

Figura 3.3 - Arquitetura de uma rede MLP com uma camada escondida e treinamento *backpropagation*

3.2.1. Treinamento da rede MLP

Conforme citado anteriormente o treinamento da rede MLP através do algoritmo *backpropagation* pode ser dividido em duas fases (*forward* e *backward*). Segundo (HAYKIN, 2001), a fase de propagação do *backpropagation* pode ser definida pelas seguintes equações:

$$z_j^{(l)} = \sum_{i=1}^{m} w_{ji}^{(l)} y_i^{(l-1)} + b_j \tag{3.3}$$

Onde $z_j^{(l)}$ é a soma ponderada de todas as entradas sinápticas (m) acrescida do bias para o neurônio j na camada l. $y_i^{(l-1)}$ é o sinal de saída do neurônio i na camada anterior $l - 1$ e w_{ji}^{l} é o peso sináptico do neurônio j da camada l que é multiplicado pela saída do neurônio i da camada anterior (l-1). Logo o sinal de saída do neurônio j na camada l pode ser dado por:

$$y_j^{(l)} = f_j(z_j) \tag{3.4}$$

Onde *f* representa a função de ativação que descreve a relação funcional de entrada e saída da não linearidade associada ao neurônio *j* na camada *l*.

As funções de ativação comumente utilizadas são a função sigmoide e tangente hiperbólica, ambas descritas na equação 3.6 e 3.7 respectivamente. Atualmente, conforme citado anteriormente uma simples função de ativação denominada ReLu vem sendo utilizada com sucesso em várias aplicações envolvendo RNAs. Testes conduzidos por (KRIZHEVSKY *et al*., 2012) reportaram uma convergência 6 vezes mais rápida das redes utilizando a função ReLu quando comparada à convergência das RNAs com função de ativação tangente hiperbólica. Sendo assim, esta é uma das funções de ativação a ser utilizada nesta tese. A equação da mesma pode ser visualizada a seguir na equação 3.8. Seja z representada na equação 3.5, as equações 3.6, 3.7 e 3.8 podem ser escritas da seguinte forma:

$$z = \sum_{i=1}^{m} w_{ji} y_i + b_j \tag{3.5}$$

$$f(z) = \frac{1}{1+e^{-(z)}} \tag{3.6}$$

$$f(z) = \tanh(z) \tag{3.7}$$

$$f(z) = \max(0, z) \tag{3.8}$$

Quando o neurônio *j* faz parte da camada de saída (L3), então o sinal de erro pode ser calculado através da equação 3.9 apresentada a seguir:

$$e_j = d_j - o_j \tag{3.9}$$

Onde d_j é o *j*-esimo elemento do vetor desejado e o_j é a saída da rede para o *j*-esimo neurônio de saída. Com o valor do erro calculado para todos os neurônios da camada de saída da rede, a função de erro a ser minimizada é dada pela equação 3.10, onde N representa o número de padrões que fazem parte do conjunto de treinamento e *C* inclui todos os neurônios existentes na camada de saída da rede.

$$E = \frac{1}{N}\left(\frac{1}{2}\sum_{j \in C} e_j^2\right) \tag{3.10}$$

Sendo assim, o objetivo do treinamento da rede MLP é ajustar os parâmetros livres da rede de tal forma a minimizar a função de erro (E) apresentada na equação 3.10. Esta

minimização pode ser realizada utilizando o método do gradiente, onde, os pesos atualizados da rede podem ser obtidos diretamente em direção contrária ao gradiente do erro, ou seja, a cada iteração do método do gradiente a adaptação dos pesos é proporcional a derivada parcial da função erro em relação aos pesos da rede multiplicada por uma taxa de aprendizagem (η) conforme equação 3.11.

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} \tag{3.11}$$

Onde o cálculo da derivada parcial ($\partial E / \partial w_{ij}$) pode ser obtido pela regra da cadeia. A equação resultante deste cálculo é a seguinte:

$$\frac{\partial E}{\partial w_{ji}} = -e_j f_j'(z_j) y_i \tag{3.12}$$

Onde f_j' é a derivada da função de ativação relacionada ao neurônio *j*, e_j e z_j estão representados nas equações 3.9 e 3.5 respectivamente. Sendo assim, pelas equações 3.11 e 3.12 a correção dos pesos sinápticos dos neurônios considerando a última camada da rede pode ser realizado com a seguinte equação:

$$\Delta w_{ji} = \eta \delta_j y_i \tag{3.13}$$

Onde Δw_{ji} é a correção a ser aplicada na *i*-esima sinapse do neurônio *j*, η é o parâmetro da taxa de aprendizagem do algoritmo, y_i é o sinal de entrada no *i*-esimo nó de entrada do neurônio *j* e o gradiente local do neurônio *j* (δ_j) é definido por $(e_j . f_j'(z_j))$.

A próxima fase do algoritmo *backpropagation* é denominada de retropropagação (*backward*) e a mesma permite a atualização dos pesos sinápticos das camadas intermediárias (ocultas). Os valores de erro encontrados na camada de saída são propagados camada por camada no sentido contrário calculando-se o gradiente local de cada neurônio e fazendo a atualização de seus respectivos pesos sinápticos. Sendo assim, a correção dos pesos sinápticos dos neurônios da camada intermediária é dada por:

$$\Delta w_{ji} = \eta(f_j'(z_j)) \sum_k \delta_k w_{kj} y_i \tag{3.14}$$

Onde $f_j'(z_j)$ depende apenas da função de ativação do neurônio *j*, $f_j'(z_j) \sum_k \delta_k w_{kj}$ é considerado como gradiente local do neurônio *j* (δ_j) e δ_k requer o conhecimento do sinal de erro para todos os neurônios na camada à direita da camada do neurônio *j*. Sendo assim, o ajuste dos pesos sinápticos em uma camada da rede para uma determinada iteração *n* é dada pela regra delta generalizada apresentada na equação 3.15 a seguir.

$$w_{ji}(n+1) = w_{ji}(n) + \beta(w_{ji}(n-1) + \eta\delta_j(n)y_i(n) \qquad (3.15)$$

Onde η é o parâmetro da taxa de aprendizagem, conforme citado anteriormente e β é a constante de momento. Uma explicação mais detalhada sobre estes parâmetros e qual a influência dos mesmos no treinamento das redes MLP pode ser encontrada em (HAYKIN, 2001).

O processo de treinamento das MLPs é um processo iterativo, sendo assim, todo o procedimento adotado anteriormente é realizado por um determinado número de iterações a ser definido no início do treinamento e o mesmo só termina quando um critério de parada definido é satisfeito.

3.3. Redes Neurais Convolucionais

As redes neurais CNNs fazem parte de um tipo particular de arquitetura de Rede Neural DLNN que possui motivação neurobiológica com origem no trabalho proposto por (HUBEL *et al*., 1962; HUBEL *et al*., 1977). Neste trabalho os autores demostraram que o córtex visual de gatos é organizado de forma hierárquica em um conjunto de células sensíveis a pequenas sub-regiões denominadas campos receptivos. De acordo com o padrão de estímulo que as células recebem, as mesmas podem ser classificadas em células simples (ativadas quando um padrão simples é apresentado ao animal), complexas e super complexas (ativadas quando padrões mais complicados ou uma combinação de padrões simples são apresentadas ao animal). Esta estrutura de um arranjo complexo de células simples e complexas responsáveis pela percepção visual humana foi então a base para as pesquisas em CNNs.

Uma CNN pode ser considerada uma rede *feedforward* com várias camadas de convolução intercaladas com camadas de subamostragem (*subsampling* ou *pooling*) e funções de ativação não linear aplicadas aos resultados e camada totalmente conectada (*fully connected)* como última camada. Fazendo uma analogia ao modelo neurobiológico citado anteriormente, as células simples são representadas no modelo pelas camadas convolucionais, nos quais exibem um mecanismo similar de conectividade local e compartilhamento de pesos. Já as células complexas combinam a ativação de células simples e são representadas na forma de camadas de *pooling*, ambas as camadas detalhadas nas próximas seções. Após o processamento destas camadas, a ativação da última camada convolucional ou *pooling* é utilizada para alimentar um modelo de classificação. Na figura 3.4 a seguir é apresentada a

arquitetura das redes CNNs estruturada em vários estágios. A título de simplificação do modelo são apresentadas apenas uma camada de convolução e uma camada de *pooling*. Entretanto, a maioria dos modelos atuais, são compostas de diversas destas camadas alternadas. O resultado gerado a partir da camada de convolução é denominado de mapas de características e os mesmos são utilizados como entrada para a próxima camada (*pooling*). Já os resultados gerados a partir da camada de *pooling* é utilizado como entrada para a camada totalmente conectada. Esta camada é responsável em gerar as saídas da rede (resultado da classificação).

Nas próximas seções todos os termos relacionados às redes CNN, assim como seus hiperparâmetros serão referenciados por seus termos em inglês, visto que é esse o modelo mais comumente utilizado na literatura, mesmo em trabalhos na língua portuguesa.

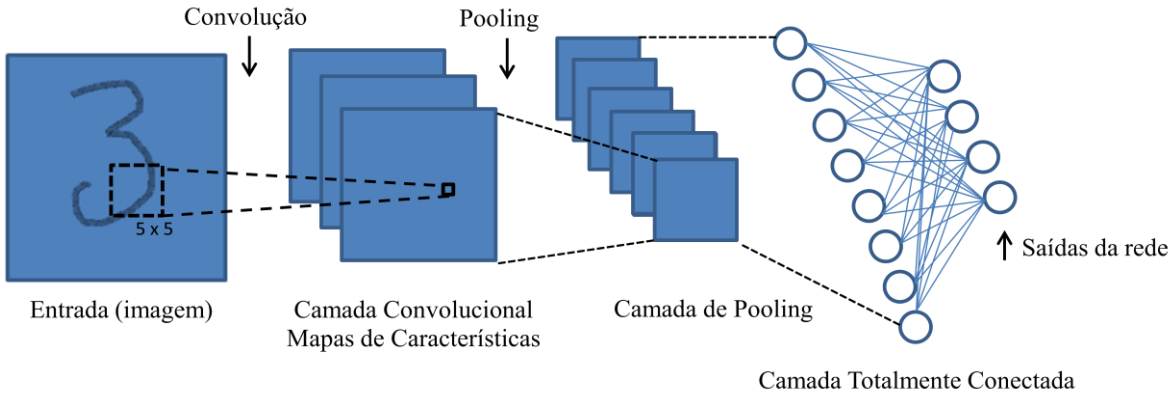

Figura 3.4 - Rede Neural Convolucional – CNN

3.3.1. Camadas Convolucionais

A camada convolucional usa filtro de aprendizado e cada filtro responde a um subconjunto ou uma pequena região do conjunto de entrada (um pequeno quadrado em uma imagem poderia ser considerado como uma pequena região) e as pequenas regiões coletivamente cobrem o conjunto de entrada como um todo. Ou seja, para cada um destes filtros, um neurônio está conectado a apenas um subconjunto dos neurônios na camada anterior. Os filtros são gerados a partir de uma pequena região denominada campo receptivo local e o resultado da aplicação destes filtros são denominados mapas de características (ou *feature maps*).

Um mapa de características é obtido efetuando a convolução de uma matriz de entrada (que em aplicações de visão computacional pode ser uma imagem ou para aplicações

envolvendo PLN podem ser *word embeddings*, como é o caso específico desta tese) por um filtro linear. A convolução é um tipo de operação linear e pode ser considerada como uma característica básica para se identificar uma CNN, ou seja, para ser considerada uma CNN é preciso que a rede neural utilize a convolução em pelo menos uma de suas camadas. O processo de convolução utilizado nas redes CNNs é apresentado na figura 3.5. Na figura, a matriz de entrada é representado por uma matriz de tamanho 5x5 (pixels de intensidade de uma imagem, por exemplo) e o filtro apresentado é um filtro específico de tamanho 3x3, denominado filtro *edge* (PARKER, 2010) que é responsável por detecção de bordas nas imagens. O objetivo dos filtros utilizados nas CNNs é detectar determinadas características nos dados de entrada, sendo assim diferentes valores de filtros podem produzir diferentes mapas de características para um mesmo tipo de dado (imagem ou texto).

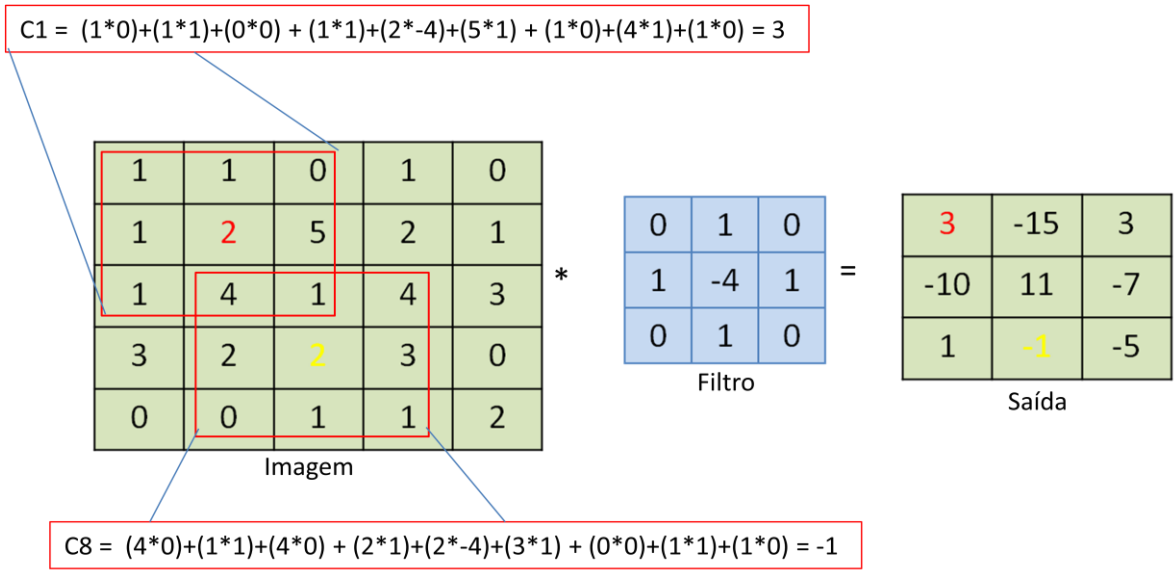

Figura 3.5 - Exemplo de Convolução 2D – A matriz de saída é denominada mapa de características (*features* maps)

Na figura 3.5 é realizada uma operação de convolução em uma imagem como entrada e um filtro que é deslizado a um pixel (hiperparâmetro da CNN conhecido como *stride* e discutido nas próximas seções) em toda a imagem. Para cada janela que desliza sobre a imagem, o produto entre cada elemento do filtro ou Kernel (também conhecido como conjuntos de pesos sinápticos) e o elemento de entrada que ele sobrepõe é calculado e os resultados são somados para obter a saída correspondente aquela localização analisada. O resultado final deste procedimento é denominado mapa de características.

As redes CNNs apresentam determinadas características que contrastam com certos paradigmas que fazem parte dos modelos de arquiteturas rasas, como as redes MLPs por exemplo. São elas: conectividade esparsa, campos receptivos locais e compartilhamento de parâmetros.

- Conectividade esparsa: este tipo de conectividade acontece devido ao fato de que o filtro (kernel) que é utilizado no processo de convolução apresenta dimensões muito menores do que a matriz de entrada. Isto permite uma correlação espacial local através da aplicação de um padrão de conectividade local entre neurônios de camadas adjacentes. Nas arquiteturas de redes MLPs, cada neurônio da camada de saída interage com cada neurônio da camada de entrada através de um processo de multiplicação de matrizes gerando uma conectividade global entre os parâmetros. Na figura 3.6 apresentada a seguir é possível verificar os dois tipos de conectividades (global e esparsa) para os dois tipos de arquiteturas (MLP e CNN). Analisando apenas a entrada x_3 é possível verificar que quando o modelo faz uso da multiplicação de matrizes (conectividade global – MLP) todas as unidades de saída (N1 a N5) são afetadas pela entrada x_3. Isto não ocorre quando o modelo faz uso da camada convolucional, neste caso utilizando um kernel de tamanho 3 por exemplo, apenas três unidades da camada de saída (N2, N3 e N4) são afetadas pela entrada x_3.
- Campos receptivos locais: cada região (conforme citado anteriormente, um pequeno quadrado em uma imagem pode ser considerado como uma região) na matriz de entrada que representa o tamanho do filtro utilizado vai corresponder a um único neurônio na camada escondida. Cada deslocamento desta região ao longo da matriz de entrada representará um novo neurônio na camada escondida. Este procedimento está representado na figura 3.7. Analisando na figura os neurônios de entrada x1 a x17 (neurônios destacados em preto na figura) e o neurônio N1 da primeira camada escondida e um filtro de convolução de tamanho 3x3 é possível verificar que somente a região destacada em preto na figura afeta o neurônio de saída N1. A esta região destacada é que se dá o nome de campo receptivo local do neurônio de saída N1. O mesmo acontece para o neurônio de saída N2.
- Compartilhamento de parâmetros: os mesmos filtros são aplicados em diferentes locais da matriz de entrada. Isto faz com que os padrões que ocorrem com frequência na matriz de entrada e que estejam localizados em qualquer parte da

entrada possam ser aprendidos (HAFEMANN et al., 2015). Na figura 3.8 pode ser visualizado como acontece o compartilhamento de parâmetros nas camadas convolucionais.

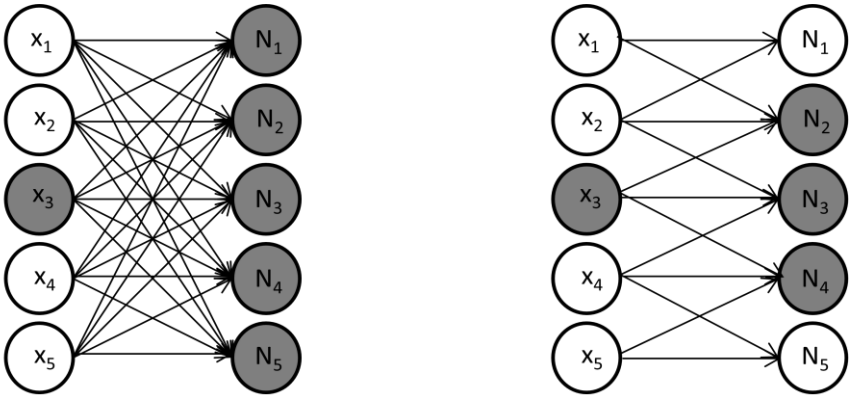

Figura 3.6 - Conectividade global e esparsa das redes MLPs e CNNs respectivamente. Adaptado de (GOODFELLOW *et al*., 2016)

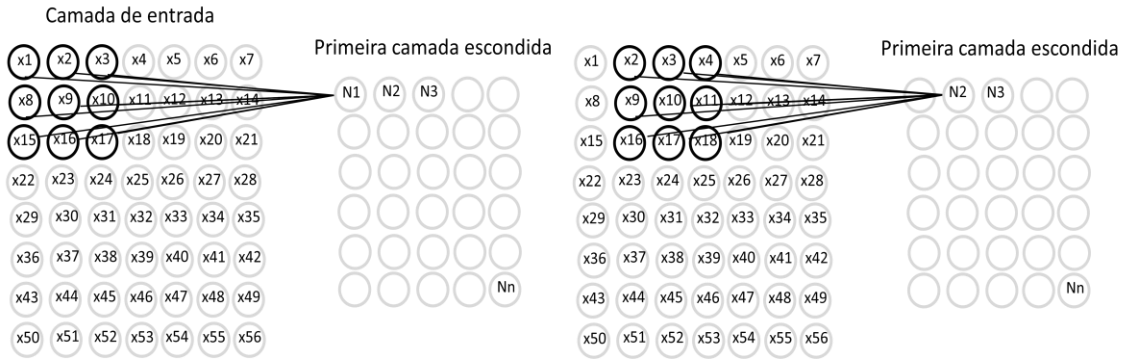

Figura 3.7 - Exemplo do campo receptivo local das CNNs. Adaptado de (NIELSEN, 2016)

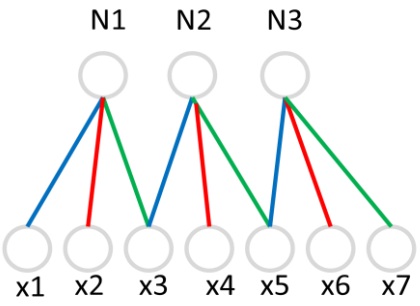

Figura 3.8 - Compartilhamento de parâmetros. Conexões na mesma cor representam pesos (ou elementos do Kernel) sendo compartilhados

Para (GOODFELLOW et al., 2016) o uso das características citadas anteriormente (conexões esparsas, campos receptivos locais e compartilhamento de parâmetros) permitem as redes CNNs armazenarem menos parâmetros, o que reduz os requisitos de memória do modelo e melhora a sua eficiência estatística. Além disso, a geração dos resultados obtidos requer menos operações, o que possibilita uma grande melhoria na eficiência destes modelos de rede.

Os resultados obtidos ao efetuar o processo de convolução em uma matriz de entrada juntamente com a adição de um termo de bias são utilizados como entrada para uma função de ativação não linear. Conforme citado anteriormente e também em (JARRETT *et al*. 2009; GLOROT *et al*., 2011; NAIR *et al*., 2010) a melhor função de ativação a ser utilizada em modelos de RNAs é a função ReLu descrita na equação 3.8, entretanto funções de ativação tangente hiperbólica (equação 3.7) e sigmoide (equação 3.6) também podem ser utilizadas tanto em modelos de rede MLP como em modelos de rede CNN.

Por fim, cabe ressaltar dois hiperparâmetros muito importantes em relação aos filtros utilizados nas camadas convolucionais, a saber, *wide* ou *narrow convolution* e o *stride*. O primeiro deles define a forma como aplicar o filtro, ou seja, dada uma matriz de dados, como aplicar o filtro aos elementos que não possuem vizinhos (elementos na borda da matriz). Uma abordagem comumente utilizada é a utilização do *padding* (preenchimento) que visa expandir matriz de entrada. Em casos onde se faz uso do *padding*, ou seja, quando há adição de elementos na borda da matriz de entrada com valor zero este hiperpârametro é conhecido *wide convolution*. O mesmo permite aplicar o filtro em todos os elementos da matriz de entrada. No caso contrário, onde não há utilização do *padding*, o hiperparâmetro é conhecido como *narrow convolution*. *Stride* (ou passo) é outro hiperparâmetro a ser definido (também pode ser utilizado nas camadas de *pooling*) e é responsável por determinar qual o tamanho do deslocamento do filtro a cada passo. Geralmente em aplicações envolvendo reconhecimento de imagens é comum o uso de *stride* igual a 1, ou seja, o filtro é deslocado pela imagem um pixel por vez.

3.3.2. Camadas de pooling

As camadas de *pooling* são usualmente utilizadas imediatamente depois das camadas convolucionais e são responsáveis por tentar encontrar a informação mais importante e significativa obtidas a partir das camadas convolucionais. Para (GOODFELLOW *et al*., 2016), as camadas de *pooling* substituem a saída das camadas convolucionais em uma

determinada região por uma estatística resumida das saídas mais próximas, o que pode gerar uma redução no número de neurônios da camada anterior. O operador de *max* é a função de *pooling* que reporta o valor máximo de uma determinada região. Outras funções também muito utilizadas são a média, a norma L2 ou a média ponderada da região local analisada.

Uma propriedade importante do *pooling* é o fato do mesmo fornecer uma matriz de saída de tamanho fixo, ou seja, citando como exemplo, uma aplicação que faz uso de 100 filtros nas camadas de convolução, a utilização do *pooling* para cada filtro resultará em um vetor saída de dimensão 100, indiferente do tamanho do filtro, ou mesmo do tamanho da entrada. Sendo assim, em aplicações envolvendo classificação de textos, onde vetores de entrada devem ter o mesmo tamanho, sentenças e filtros de diferentes tamanhos resultarão sempre em um vetor saída de mesma dimensão, possibilitando assim o uso do classificador. Em aplicações envolvendo reconhecimento de imagens, o *pooling* também fornece invariância a translação e rotação, ou seja, indiferente da operação realizada na imagem, o operador de *max* escolherá sempre o mesmo valor (valor máximo da região analisada). A figura 3.9 a seguir apresenta o operador de máximo (*max*) para uma matriz de saída de tamanho 4x6 com uma janela de tamanho 2x2.

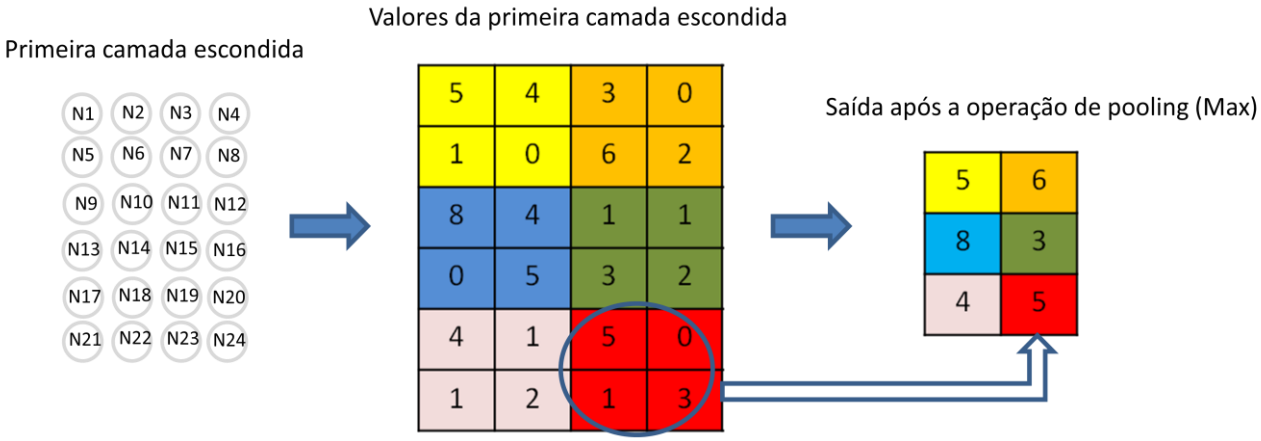

Figura 3.9 - *Pooling* usando operador Max, com filtro 2x2 e *stride* igual a 2

É possível visualizar através da figura apresentada anteriormente (figura 3.9) que a saída da camada convolucional é composta de 4 x 6 neurônios e que após o processamento da camada de *pooling* restou apenas uma saída de 2 x 3 neurônios. Uma redução significativa, onde somente as melhores características ficam armazenadas. E consequentemente com a redução do número de características, há também uma redução no número de parâmetros necessários nas camadas posteriores.

3.3.3. Dropout

Conforme citado anteriormente as redes CNNs consistem de camadas alternativas de convolução e camadas de *pooling*, com camadas totalmente conectada (*fully connected*) finalizando o modelo. Se compararmos as redes CNNs com as redes MLPs tradicionais (arquiteturas rasas) é possível confirmar que as CNNs apresentam muito menos conexões e parâmetros devido a presença de conectividade local e armazenamento de filtros (para detalhes ver seção 3.3.1). Entretanto as aplicações que fazem uso deste tipo de arquitetura podem chegar a apresentar milhares de camadas de processamento aumentando substancialmente a quantidade de parâmetros da rede. Consequentemente, esta maior complexidade da CNN pode apresentar um problema conhecido como sobreajuste (*overfitting*), ou seja, quando o modelo se ajusta muito aos dados apresentados durante o processo de treinamento e consequentemente não generaliza bem.

Na figura 3.10 pode ser visualizada a ocorrência do *overfitting* no treinamento das Redes Neurais Artificiais (por exemplo: CNN). No gráfico superior é apresentado o erro de treinamento onde o critério de parada do algoritmo ocorre de forma tardia (ocorrência do *overfitting*). É possível constatar que o erro nos dados de treinamento continua diminuindo, entretanto, o erro do conjunto de teste começou a aumentar. Já no gráfico inferior, com o uso das técnicas de regularização, é possível constatar que o treinamento é encerrado em um ponto ótimo de generalização, ou seja, antes que o erro no conjunto de teste aumentasse.

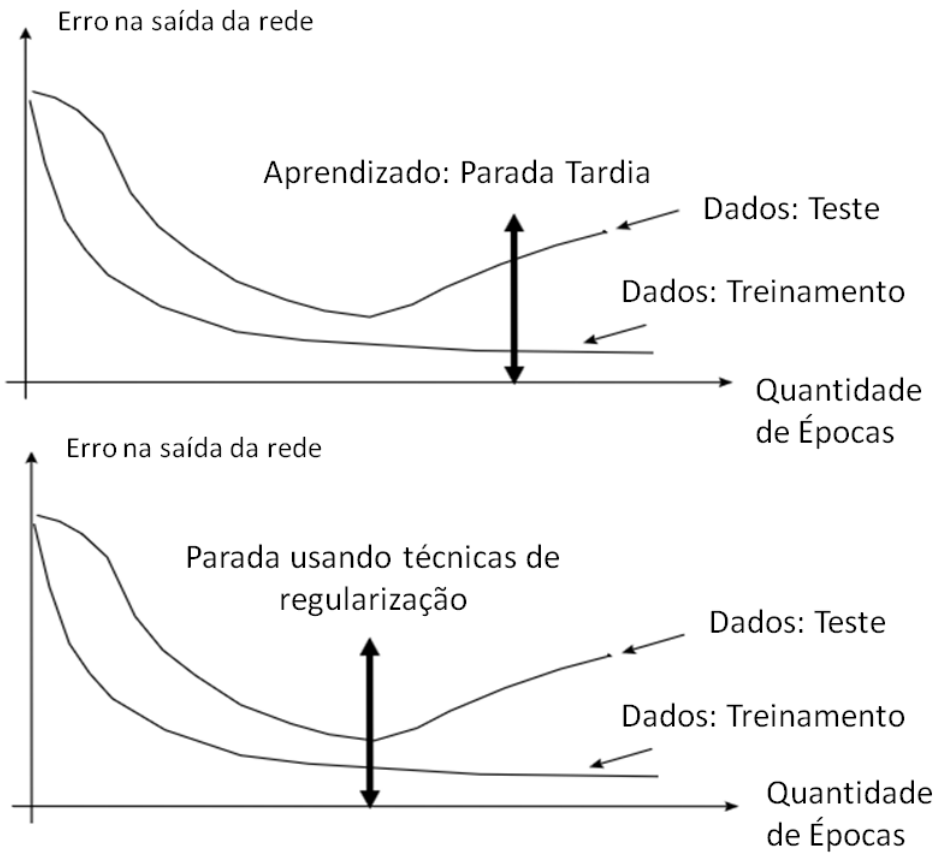

Figura 3.10 - Problema de *overfitting* que ocorre durante o treinamento das Redes Neurais Artificiais

O *overfitting* pode ser resolvido através de diferentes técnicas de regularização. Estas técnicas são responsáveis em realizar uma modificação no algoritmo de aprendizagem a fim de reduzir o erro de generalização, mas não necessariamente o erro do treinamento. Sua eficácia pode ser medida por um bom *trade-off* entre a redução de variância e o aumento de viés de estimadores (GOODFELLOW *et al*., 2016).

Uma das técnicas de regularização que vem sendo utilizada com sucesso em aplicações envolvendo a DLNN é conhecida como *dropout*. A mesma foi proposta por (HINTON *et al*., 2012) e visa definir estocasticamente zeros a saída de alguns neurônios de uma determinada camada com probabilidade de p= 0.5. O restante dos pesos dos neurônios não modificados são treinados pelo algoritmo *backpropagation*. O procedimento é repetido para cada exemplo em cada época de treinamento. Um exemplo deste procedimento pode ser visualizado na figura 3.11.

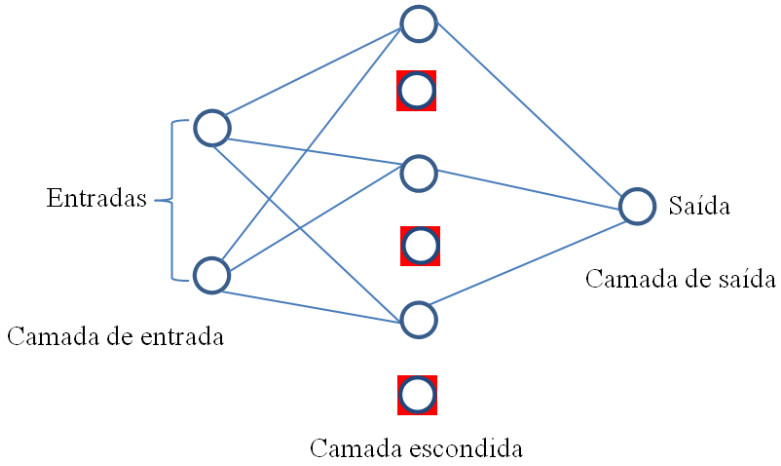

Figura 3.11 - Operação de *dropout* para uma rede neural simples com apenas uma camada escondida. Adaptado de (BALDI *et al*. 2014)

Conforme a figura 3.11, para cada exemplo de treinamento (entradas), os neurônios da camada escondida (marcados em vermelho) e suas respectivas conexões são descartados temporariamente com probabilidade de p=0,5. Os demais neurônios têm seus pesos ajustados normalmente através do algoritmo *backpropagation*. A ideia é que existe uma arquitetura de rede diferente a cada vez que um exemplo de treinamento é apresentado, mas todas essas redes compartilham os mesmos pesos para as unidades escondidas que estão presentes.

A técnica de *dropout* pode ser utilizada em qualquer camada da rede DLNN, podendo inclusive ser também aplicada à camada de entrada desativando aleatoriamente alguns dos componentes do vetor de entrada. Neste caso, geralmente é feito o uso de um valor de probabilidade menor, p=0,2 (BALDI *et al*., 2013).

HINTON *et al*. (2012) afirmaram que a técnica de *dropout* pode reduzir as co-adaptações complexas de neurônios, uma vez que os neurônios da camada associada desativados não podem influenciar os demais neurônios. Consequentemente os neurônios aprendem características mais robustas aumentando assim a eficiência na generalização do modelo. Os autores afirmaram também que a técnica além de ser eficiente para reduzir o erro no conjunto de testes (generalização), a mesma é computacionalmente eficiente pois é capaz de treinar um grande número de redes diferentes em um tempo razoável. O efeito da abordagem deste algoritmo é similar ao uso de múltiplos modelos de redes e a avaliação dos mesmos em cada exemplo de teste. Entretanto isto pode ser impraticável, uma vez que o treinamento e a avaliação de tais modelos são dispendiosos em termos de execução e memória.

3.3.4. Camada totalmente conectada

As redes CNNs apresentam uma ou mais camadas totalmente conectadas (*fully connected*). O objetivo desta camada é usar as características (ou mapas de ativação) de mais alto nível que são geradas a partir das camadas anteriores (camadas convolucionais e *pooling*) para fazer a classificação dos dados de entrada (imagem, textos ou som) em várias classes baseada em um conjunto de treinamento.

Diferentemente das camadas anteriores, onde os pesos são conectados a apenas uma determinada região, a camada totalmente conectada possui a característica de ser completamente conectada, uma vez que, cada neurônio da camada anterior (geralmente a camada de *pooling*) é conectado a cada um dos neurônios da próxima camada (primeira camada *fully connected*). A mesma é composta também de uma última camada, denominada camada de saída, onde cada neurônio representa uma determinada classe (alvo) do modelo, sendo assim, o número de neurônios da camada de saída é correspondente a quantidade de classes presentes no modelo.

Na figura 3.12 pode ser visualizada esta estrutura, onde é possível observar 5 mapas de características de tamanho 4 x 4 (resultados dos processamentos das camadas anteriores – convolucional ou *pooling*), onde cada elemento da matriz de saída é conectado a um determinado neurônio de entrada. Estes resultados são colocados de forma linear formando a primeira camada totalmente conectada (camada *flatten* composta por 80 neurônios). E consequentemente estas entradas vão alimentar a segunda camada totalmente conectada (camada escondida composta por 7 neurônios). No final da cadeia de processamento da camada totalmente conectada pode ser visualizada a camada de saída (composta por 3 neurônios) que correspondem as classes do modelo. Cabe ressaltar que se for analisada apenas a camada totalmente conectada é possível identificá-la como uma rede neural tradicional MLP.

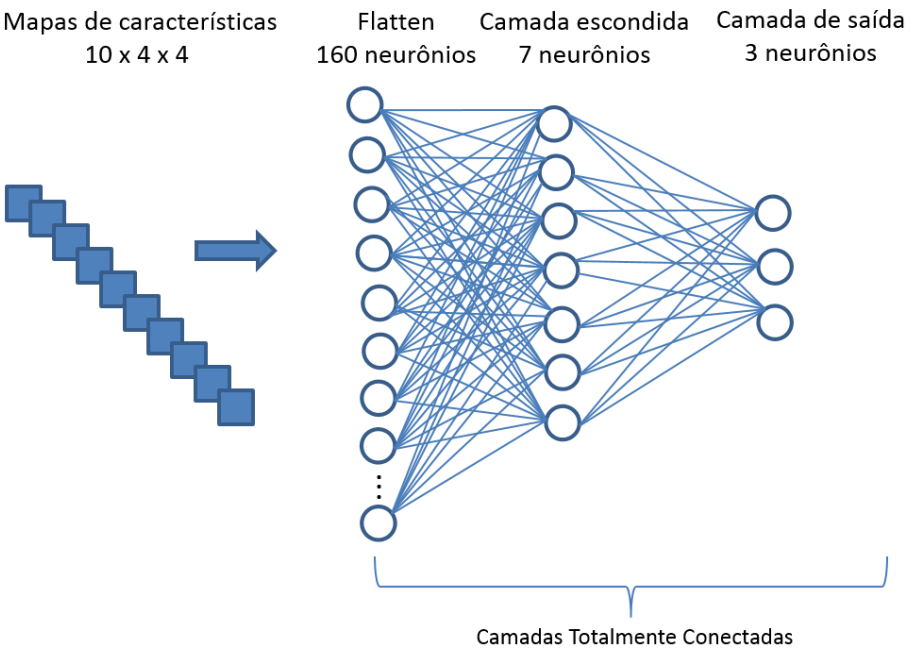

Figura 3.12 - Camadas totalmente conectada composta por 3 camadas (*flatten*, escondida e saída)

O treinamento das redes CNNs é realizado utilizando o algoritmo *backpropagation* detalhado na seção 3.2.1 que visa minimizar a diferença entra a saída da CNN e a saída desejada. O treinamento é um processo iterativo e só termina quando um critério de parada é satisfeito.

A função de ativação utilizada na camada de saída da camada totalmente conectada é a função *softmax* descrita na equação 3.16, onde Y corresponde ao vetor de saída do classificador. Esta função transforma um vetor de pontuação arbitrária de valores reais em um vetor de probabilidades entre zero e um e que juntos somam 1. E a resposta final do classificador pode ser obtida através da seleção do item que possui a maior probabilidade.

$$soft\max(y_i) = \frac{e^{z_i}}{\sum_{j \in Y} e^{z_j}} \tag{3.16}$$

3.4. Redes Neurais Máquinas de Aprendizado Extremo

Os resultados obtidos a partir do processamento das camadas de convolução e *pooling* da rede CNN, ou seja, a camada (*fully connected)* passam por um classificador não linear que

permite a rede prever uma determinada classe dada uma palavra ou sentença como entrada. Conforme citado anteriormente, a grande maioria dos trabalhos citados na literatura fazem uso de uma arquitetura rasa (como a rede neural MLP por exemplo) com algoritmo de treinamento *backpropagation* ou mesmo o algoritmo SVM. A proposta desta tese é fazer o uso de um classificador diferente dos comumente utilizados na literatura, uma arquitetura de rede neural denominada Máquinas de Aprendizado Extremo (*Extreme Learning Machin*e – ELM).

A ELM consiste de uma rede *feedforward* com uma única camada oculta, proposta por (HUANG *et al*., 2004; HUANG *et al*., 2006) podendo ser utilizadas em aplicações envolvendo classificação de dados e regressão. Conforme citado na seção 3.2.1 a forma tradicional de se treinar uma rede *feedforward* é através do algoritmo *backpropagation* que consiste em ajustar os pesos sinápticos das camadas ocultas e de saída da rede de forma a minimizar o erro entre a saída da mesma e a saída real do conjunto de treinamento. Entretanto a ELM apresenta um caso especial das arquiteturas *feedforward*, pois além de apresentarem somente uma camada escondida, as mesmas também são treinadas por um algoritmo diferente do comumente utilizado.

Na rede ELM os pesos entre as camadas de entrada e oculta são escolhidos aleatoriamente, sem a necessidade de treinamento, ou seja, os mesmos não são alterados até o final do algoritmo. Apenas os pesos dos neurônios entre a camada oculta e saída, são determinados analiticamente, utilizando a inversa generalizada de *Moore-Penrose*, ou pseudo inversa (SERRE, 2002). Uma estrutura da rede ELM pode ser visualizada na figura 3.13.

Em (HUNG *et al*., 2011), os autores comprovaram formalmente que a capacidade de aprendizagem da ELM pode ser avaliada do ponto de vista de dois aspectos: a capacidade de interpolação e a capacidade de aproximação universal. Do ponto de vista de interpolação, os autores afirmaram que uma variedade grande de funções de ativação pode ser utilizada com a rede ELM, como a função sigmoide, função de base radial, tangente hiperbólica, seno, exponencial, cosseno e muitas outras não regulares (HUANG *et al*., 1998). Os autores afirmaram também que a maioria das funções de ativação comumente utilizadas satisfazem as condições que garantem a capacidade de aproximação universal da ELM.

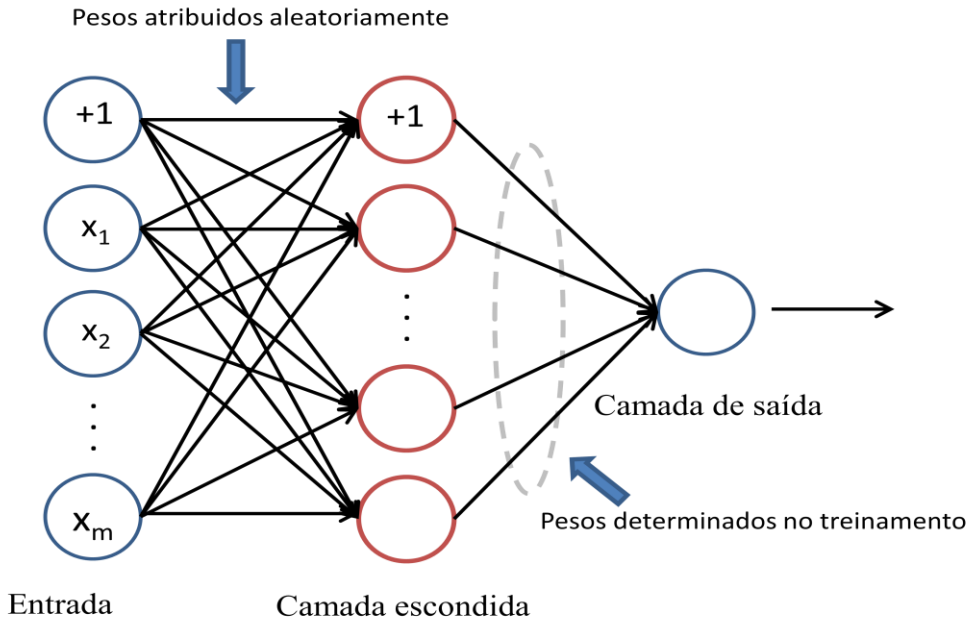

Figura 3.13 - Arquitetura ELM composta de três camadas (entrada, escondida e camada de saída)

3.4.1. Treinamento da rede ELM

Conforme citado anteriormente o treinamento da rede ELM é realizado de forma analítica diferentemente do algoritmo *backpropagation* que possui uma abordagem iterativa. O objetivo do treinamento da ELM é encontrar uma matriz de pesos β baseado na saída T (vetor alvo) e na matriz de saída da camada escondida (H) através da resolução de um sistema linear. Sendo assim, para uma rede com p unidades de entrada, q neurônios ocultos e C saídas, a i-esima saída na iteração k é definida por:

$$o_i(k) = \beta_i^T h(k) \tag{3.17}$$

Onde $\beta_i \in \mathbb{R}^q$, $i = 1,....,C$, é o vetor de pesos conectando os neurônios ocultos ao i-esimo neurônio de saída, e $h(k) \in \mathbb{R}^q$ é o vetor de saídas dos neurônios ocultos para um dado padrão de entrada $a(k) \in \mathbb{R}^p$. O vetor *h(k)* propriamente dito é dado como:

$$h(k) = \left[f(W_1^T a(k) + b_1),..., f(W_q^T a(k) + b_q) \right]^T \tag{3.18}$$

Onde b_l , $l = 1....q$ é o bias do l-esimo neurônio oculto, $W_l \in \mathbb{R}^p$ é o vetor de pesos do l-esimo neurônio oculto e *f(.)* é a função de ativação. Conforme citado anteriormente várias funções de ativação podem ser utilizadas com a rede ELM, entretanto, neste trabalho em específico é abordada apenas a função sigmoide.

Seja H = [h(1) h(2) ... h(*N*)], uma matriz *q x N* cujas *N* colunas são os vetores de saída da camada escondida $h(k) \in \mathbb{R}^q$, k = 1,...*N*, onde N é o número de padrões de entrada de treinamento disponíveis. Da mesma forma, seja T = [t(1) t(2) ... t(*N*)], uma matriz *C x N* cuja *k*-esima coluna é o vetor alvo $t(k) \in \mathbb{R}^C$, associado com o padrão de entrada a(*k*), k = 1 ... *N*. E por fim, seja $\beta = [\beta_1\ \beta_2 \dots \beta_C]$ uma matriz *q x C*, cuja *i*-esima coluna é o vetor de pesos $\beta_i \in \mathbb{R}^q, i = 1 \dots, C$. Portanto, estas três matrizes estão relacionadas pelo mapeamento linear T = $\beta^T H$, onde as matrizes T e H são conhecidas e a matriz de pesos β pode ser facilmente calculada por meio do método da pseudoinversa conforme equação 3.19.

$$\beta = (HH^T)^{-1} HT^T \tag{3.19}$$

Diferentes métodos podem ser utilizados para se fazer o cálculo da matriz inversa generalizada de *Moore-Penrose*, dentre os quais é possível destacar: método da projeção ortogonal, métodos interativos, decomposição de valor singular (SVD) e método de ortogonalização (RAO *et al*., 1972).

A fim de melhorar a performance de generalização dos modelos que fazem uso da rede ELM é possível fazer o uso de algumas técnicas de regularização. A facilidade de se modificar os modelos de ELM possibilitam a inclusão de termos de regularização. Sendo assim, possíveis modificações na estimativa de pesos da camada escondida têm sido propostas nos últimos anos. HUNG *et al*. (2012) apresentaram uma versão onde um termo de regularização *C* foi adicionado ao cálculo da pseudoinversa. Sendo assim em casos onde o número de amostras de treinamento (N) seja menor que a quantidade de neurônios da camada escondida (N < *q*) a matriz de pesos β pode ser encontrada pela equação 3.20.

$$\beta = H\left(\frac{I}{C} + HH^T\right)^{-1} T^T \tag{3.20}$$

Caso contrário, ou seja, se N > q (o que geralmente acontece), o cálculo de β pode ser realizado pela equação 3.21.

$$\beta = \left(\frac{I}{C} + H^T H\right)^{-1} HT^T \tag{3.21}$$

Onde I é uma matriz identidade de dimensão *q* e C é uma constante positiva para controlar o grau de regularização.

Em (MARTÍNEZ-MARTÍNEZ *et al*., 2011), os autores consideraram algumas regularizações para regressão de mínimos quadrados para a determinação dos pesos da camada de saída da ELM. Foram abordados os seguintes métodos: lasso (TIBSHIRANI,

2011), regularização *ridge* (HOERL, *et al*., 2000) e o *elastic net* (ZOU *et al*., 2005). Em (DENG *et al*., 2009) foi proposta uma ELM regularizada que consiste na minimização do risco estrutural e no princípio da teoria da aprendizagem estatística. A formulação desenvolvida pelos autores introduz um fator de peso γ que permite ajustar a proporção do risco empírico e do risco estrutural. Além disso, as variáveis de erro ε_i podem ser ponderadas pelos pesos v_i obtendo assim uma estimativa mais robusta que enfraquece a interferência de *outliers* (dados discrepantes das amostras regulares que geralmente ocorrem nos dados de treinamento e que podem ser adquiridos durante o processo de aquisição dos dados). Outros métodos de regularização também podem ser encontrados em (HUANG *et. al*., 2011).

3.4.2. Características da ELM

A ELM oferece características significativas que vem chamando a atenção de pesquisadores na área de Inteligência computacional e comunidades de aprendizado de máquina, envolvendo tanto estudos teóricos como também na solução de problemas em aplicações de regressão e classificação (HUANG *et. al*., 2011).

Uma das principais características da ELM está na velocidade do aprendizado. Por não apresentarem um processo iterativo, como no caso de outros algoritmos de treinamento (como por exemplo o algoritmo *backpropagation* e Máquinas de Vetor de Suporte para regressão - SVR) o tempo necessário para o treinamento das redes ELMs geralmente é significativamente menor quando comparados aos algoritmos citados. De fato, (DENG *et al*., 2009) reportaram um estudo onde foi feita uma comparação entre a velocidade de treinamento da ELM em relação ao treinamento da rede MLP utilizando o algoritmo *backpropagation* e o SVR. Os autores constataram que a velocidade de treinamento da ELM é centena de vezes mais rápida que os dois algoritmos analisados.

Outra característica importante da ELM é uma boa capacidade de generalização. Em (HUANG *et. al*., 2006), os autores fizeram uma comparação envolvendo as redes ELMs e algoritmos de aprendizados citados anteriormente (*backpropagation* e SVR) em problemas de aproximação de funções. Oito problemas de aproximação de funções que fazem parte da base de dados UCI[1] foram utilizados neste estudo. Os autores reportaram que o desempenho da generalização das redes ELMs é muito similar ao encontrado pelo SVR na maioria dos casos

[1] Disponível em: http://mlearn.ics.uci.edu/MLRepository.html

e que em relação ao *backpropagation*, as redes ELMs apresentaram uma generalização muito melhor em 4 das bases de dados analisadas e nas demais o desempenho de generalização foi praticamente o mesmo.

A simplicidade na configuração das ELMs também é uma característica a ser destacada, pois diferentemente dos algoritmos tradicionais de aprendizado de máquina onde se faz necessário a configuração de muitos parâmetros de aprendizagem, nas redes ELMs o único parâmetro a ser ajustado é a quantidade de neurônios da camada escondida.

3.4.3. ELM e suas variações

A facilidade na modelagem das redes ELMs tem permitido a implementação de diversas versões das mesmas. Dentre estas versões é possível destacar o trabalho proposto por (LIANG *et. al*., 2006), onde foi apresentada uma versão online da ELM denominada *Online Sequencial Extreme Learning Machine* (OS-ELM). A justificativa dos autores para a implementação desta nova versão se refere ao fato de que muitas das aplicações que fazem uso das redes ELMs usam o método de treinamento de aprendizado em lote. E que este tipo de aprendizagem consome muito tempo, pois envolve muitas iterações através dos dados de treinamento. Ademais, sempre que um novo dado é recebido, o aprendizado em lote usa os dados passados junto com os novos dados, o que gera um retreinamento dos dados consumindo assim muito tempo. Dentro deste contexto, o algoritmo proposto visou uma modificação no algoritmo de treinamento original da ELM de forma a permitir a inclusão de novos padrões de treinamento de forma online e sequencial, sem a necessidade de se repetir os cálculos realizados anteriormente para os pesos da camada de saída. De forma resumida, as modificações propostas pelos autores foram as seguintes:

1. As observações de treinamento são apresentadas ao algoritmo de aprendizagem sequencialmente podendo ser uma a uma ou bloco a bloco (comprimento fixo ou varável).
2. A qualquer momento, apenas o dado recém-chegado (seja único ou bloco) do conjunto de observações são processados e aprendidos.
3. Assim que o procedimento de aprendizagem para um determinado dado (único ou bloco) é completada, o mesmo é descartado do processo de treinamento

4. Diferentemente do aprendizado em lote, o algoritmo de treinamento não tem conhecimento prévio de como a grande quantidade de observações de treinamento serão apresentadas.

Por fim, os autores afirmaram que a performance obtida pelo método proposto quando comparados a outros métodos de aprendizado sequencial envolvendo problemas de *benchmark* de regressão, classificação e previsão de séries temporais foram muito superiores, produzindo uma melhor generalização a uma velocidade de aprendizagem muito mais rápida.

A partir das análises e dos resultados promissores encontrados pelos autores no trabalho citado, cabe salientar que esta foi a abordagem (OS-ELM) adotada nesta tese, ou seja, o OS-ELM foi o classificador utilizado para a tarefa de classificação de notícias e previsão do mercado financeiro brasileiro, principal motivo da tese proposta.

Uma outra variação das redes ELMs também muito citada na literatura é o modelo proposto por (ZHU *et. al*., 2005) denominado *Evolutionary Extreme Learning Machine* (E-ELM). O modelo proposto pelos autores faz uso do algoritmo Evolucionário Diferencial (*Diferencial Evolution* - DE) que é conhecido por sua capacidade e eficiência na localização de um ótimo global conforme citado em (STORN *et. al*., 1997). O objetivo do modelo é encontrar um conjunto ótimo dos pesos e bias da camada de entrada da rede, descartando neurônios que tem pouca relevância para o modelo. A modificação apresentada pelos autores possibilita a obtenção de uma rede mais compacta e uma melhora no tempo de teste da rede. O processo de seleção de neurônios no modelo é realizado através de uma função de aptidão (*fitness function*) onde o cálculo é feito pela Raiz do Erro Médio Quadrático (*Root Mean Square Error* - RMSE) sobre um conjunto de validação. Uma vez que a aptidão de todos os neurônios (indivíduos na população) é calculada, são aplicados três passos que fazem parte do algoritmo DE: mutação, *crossover* e seleção. Ao final do processo de seleção, os autores visando uma melhoria no processo de generalização do modelo adicionaram um novo critério de seleção: a norma dos pesos de saída. Deste modo quando a diferença entre as aptidões de diferentes indivíduos é pequena, o indivíduo que possui a menor norma dos pesos de saída é selecionado. Este processo é repetido até que um critério de parada seja alcançado.

Em (HUANG *et al*., 2006) foi proposto o modelo *Incremental Extreme Learning Machine* (I-ELM) que possibilitou a implementação incremental de redes ELMs. O algoritmo pode ser resumido da seguinte forma: inicialmente alguns parâmetros precisam ser informados como a quantidade máxima de neurônios, a função de ativação e o erro esperado. A cada passo do algoritmo um novo neurônio é adicionado à camada escondida. Para o peso e bias que conectam este neurônio às entradas são atribuídos valores aleatórios. Já em relação

ao peso e bias que ligam o novo neurônio às saídas os valores dos mesmos são estimados segundo (KWOK *et al*., 1997). Em seguida o erro é calculado e um novo neurônio é adicionado à camada escondida. O treinamento pode ser finalizado quando é alcançado o número máximo de neurônios ou quando o erro esperado é encontrado, ambos os parâmetros informados no início do algoritmo.

Outras variações das redes ELMs também podem ser encontradas em (LAN *et al*., 2009; DING *et al*., 2015 e HUANG *et al*., 2011).

4. Processamento de Linguagem Natural e Representação das palavras

O processamento de dados textuais ou dados não estruturados que se apresentam em linguagem natural é realizado de forma bem diferente que o processamento de dados em visão computacional (imagens) ou qualquer outro tipo de dado que fazem uso dos algoritmos de aprendizagem de máquinas. Fazer com que os computadores consigam analisar, compreender e obter conhecimento a partir da linguagem humana de uma maneira útil e inteligente é uma tarefa da área denominada de processamento de linguagem natural (PLN). Através da PLN é possível organizar e estruturar o conhecimento obtido em dados textuais para então realizar tarefas como o resumo automático, tradução automática, reconhecimento de entidade nomeada (*named entity recognition*), extração de relacionamento, análise de sentimento, classificação e agrupamentos de textos e ou documentos, segmentação morfológica, etc.

Embora existam alguns trabalhos envolvendo a área de PLN antes dos anos 50, muitos consideram que os trabalhos relacionados a área de PLN só tiveram início nos anos 50 com o famoso artigo de Turing (TURING, 1950) cujo objetivo foi determinar se as máquinas poderiam pensar como seres humanos. No experimento original de Turing, um juiz conversa em linguagem natural escrita, com outro humano e com um computador em tempo real, sem o conhecimento de quem é o computador e quem é o ser humano. Ao final da conversação, apenas com base no conteúdo escrito, se o juiz não fosse capaz de distinguir quem é o humano, então segundo o teste de *Turing* conclui-se que o computador pode pensar.

O próximo trabalho que revolucionou a área de PLN ainda nos anos 50 foi proposto por (REYNOLDS, 1954) com foco na tradução automática. Nos anos seguintes houve um grande entusiasmo da comunidade científica e muitos sistemas envolvendo PLN foram implementados como o *Shrdlu* (WINOGRAD, 1971), *Eliza* (WEIZENBAUM, 1976), *Parry* (COLBY, 1981), entre outros. Apesar deste entusiasmo e do fato destes sistemas terem influenciado a implementação de muitos outros nos anos seguintes, havia uma preocupação com o conjunto complexo de regras escritas a mão que eram necessárias na implementação dos mesmos. Somente com a introdução dos algoritmos de aprendizagem de máquina, já no final dos anos 80, a área de PLN realmente teve um grande avanço permitindo assim a implementação de sistemas mais robustos e que pudessem ser utilizados em diferentes tarefas nas quais somente os humanos possuíam capacidade para realizá-las.

4.1. Representação das palavras

O primeiro passo e indiscutivelmente o mais importante em tarefas de PLN é a forma como podemos representar as palavras de modo que os algoritmos computacionais possam entendê-las e assim extrair informações a partir das mesmas. A forma mais comum para fazer esta representação é através do Modelo de Espaço Vetorial (*Vector Space Model* – VSM) desenvolvido por (SALTON *et al*., 1975), que visa resolver os problemas de representação dos documentos através da representação geométrica.

No modelo VSM os documentos podem ser representados como pontos (ou vetores) em um espaço *t*-dimensional, onde *t* é o número de termos ou palavras diferentes existentes na coleção de documentos analisada e cada dimensão (t_1, t_2 ..., t_m) corresponde a uma palavra do vocabulário (REZENDE, 2003). Pode-se dizer então que o *i*-esimo componente do vetor documento expressa o número de vezes que a palavra com o índice *i* aparece em um determinado documento. Cada palavra também pode ter um peso associado que descreve a sua significância. Os pontos que estão próximos entre si neste espaço *t*-dimensional podem ser considerados como semanticamente similares, sendo assim é possível obter por exemplo, a similaridade entre documentos a partir do cálculo da distância entre os pontos ou utilizando a função *cosine vector similarity* conforme equação 4.3.

Uma das principais tarefas em PLN é a recuperação de informações (RI), que é basicamente uma tarefa onde o sistema responde a uma consulta (*query*) fornecida pelo usuário. É uma das tarefas de PLN mais utilizadas atualmente devido ao crescimento dos mecanismos de busca na *web* (do inglês: *web search engines*) como o *Google*, por exemplo. Geralmente os sistemas de RI funcionam da seguinte forma: existe uma coleção de documentos e os mesmos podem ser páginas da *web*, conforme citado anteriormente ou artigos da literatura em forma digital ou mesmo arquivos de textos armazenados em computadores pessoais. Em seguida um usuário digita uma *query* ao sistema para acionar o mecanismo de busca e a resposta retornada são documentos relevantes ao usuário. Na figura 4.1 apresentada a seguir pode ser visualizada a forma como é abordado este tipo de sistema de RI em uma representação vetorial. Através da figura é possível verificar um espaço de alta dimensão onde cada dimensão corresponde a um termo (t_1, t_2 ..., t_3). E é possível visualizar também que vários outros vetores foram considerados neste espaço tridimensional, ou seja, n vetores representando os documentos e um vetor representando um documento de busca ou *query*. Dentro deste contexto de representação vetorial pode-se medir então a similaridade

entre a *query* e todos os demais vetores de documentos ou mesmo entre os termos (t_1, t_2 ..., t_3) e os demais vetores de documentos.

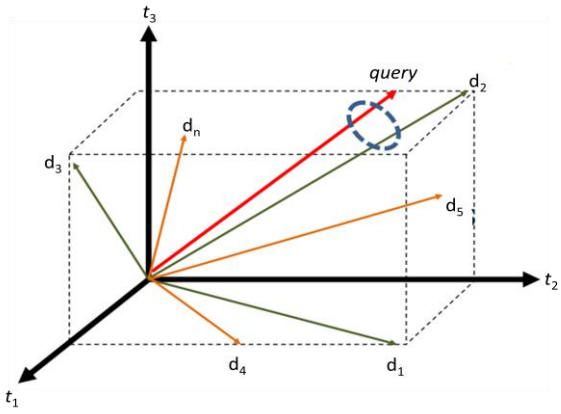

Figura 4.1 - Representação gráfica do modelo VSM

Analisando então o exemplo citado na figura 4.1 é possível verificar que o documento d_1 está mais próximo dos termos t_1 e t_2 do que do termo t_3, assim como o vetor *query* está mais próximo do documento d_2.

O modelo mais simples da representação VSM é o vetor denominado *one-hot*, ou seja, um vetor esparso do tamanho do vocabulário com uma representação binária que define valores de peso 0 e 1, ou seja, 1 quando a palavra pertence ao documento e 0 quando não pertence. Na figura 4.2 é apresentada uma matriz de *n* documentos com 6 termos e seus respectivos pesos (0 e 1).

	t_1	t_2	t_3	t_4	t_5	t_6
d_1	1	0	1	1	0	0
d_2	1	0	1	0	1	0
⋮						
d_n	0	1	0	1	0	0

Figura 4.2 - Representação VSM (*one-hot*)

Uma versão ponderada dos pesos também pode ser utilizada nos modelos VSM através do cálculo da frequência do termo (*term frequency - tf*), frequência inversa do documento (*inverse document frequency - idf*) ou até mesmo pela combinação dos dois cálculos anteriores (*tfidf*). A frequência do termo tf_{ij} está relacionada a frequência do termo *i*

em um documento *j*. Já o cálculo da *idf* e a *tfidf* podem ser realizados utilizando as equações 4.1 e 4.2 respectivamente.

$$idf_i = \log(\frac{N}{df_i}) \tag{4.1}$$

Onde *N* é a quantidade de documentos a ser analisado e df_i representa o número de documentos onde o termo *i* aparece.

$$tfidf_i = tf_{ij} \log(\frac{N}{df_i}) \tag{4.2}$$

Conforme citado anteriormente, o modelo VSM permite o cálculo da similaridade entre documentos, sendo assim, considerando um espaço vetorial contendo *m* dimensões, a similaridade (*sim*) entre um documento d_j e um outro documento (de busca) d_q pode ser calculada através do cosseno do ângulo entre os vetores dos dois documentos, ou seja, pelo produto interno dos vetores após a normalização dos mesmos para o comprimento da unidade, conforme equação 4.3. Várias outras medidas geométricas populares também podem ser utilizadas para o cálculo da similaridade entre vetores documentos ou para outras tarefas de PLN, como por exemplo, a distância euclidiana e a distância de *Manhattan*, assim como a medida de *Hellinger, Bhattacharya e Kullback-Leiber*. Entretanto, (TURNEY *et al*., 2010) afirma que o uso do cosseno do ângulo permite melhores resultados principalmente quando são utilizados vetores de documentos similares, porém com valores de frequência dos termos muito diferentes, pois o mesmo não leva em consideração o tamanho dos vetores e sim o ângulo entre os mesmos.

De fato, no trabalho de (BULLINARIA *et al*., 2007), os autores fizeram uma comparação das cinco medidas de distância citadas anteriormente juntamente com a medida de similaridade do cosseno em 4 tarefas diferentes de PLN envolvendo a similaridade entre os termos. E os mesmos afirmaram que de uma maneira geral é possível concluir que a medida do cosseno foi a que apresentou melhores resultados.

$$sim(d_j d_q) = \frac{\sum_{i=1}^{m}(w_{ij}.w_{iq})}{\sqrt{\sum_{i=1}^{m} w_{ij}^2}.\sqrt{\sum_{i=1}^{m} w_{iq}^2}} \tag{4.3}$$

Onde w_{ij} representa o peso do *i*-ésimo termo do documento d_j e w_{iq} representa o peso do *i*-ésimo termo do documento d_q.

A soma de todos os vetores *one-hot* apresentado na figura 4.2 é conhecido com um vetor de características denominado de hipótese “saco de palavras” (*Bag-of-Words* -BOW),

que apresenta valores não zeros (1 ou valores da frequência do termo - *tf, idf, tfidf*) para toda a entrada que ocorreu no documento. Nesta hipótese os documentos podem ser vistos como "*containers*" de palavras, onde cada linha representa um documento (d_n) e cada coluna representa um termo (t_m). A ordem em que as palavras aparecem nos textos são ignoradas, sendo assim, nenhum relacionamento semântico entre as palavras é analisado, ou seja, o vetor não tenta capturar a estrutura nas frases, sentenças e parágrafos do documento. Pode se dizer então que esta abordagem é visivelmente pobre em relação a todos os recursos que o vocabulário de uma língua pode oferecer, inviabilizando muitas técnicas de PLN.

Outro problema na utilização da BOW está na alta dimensionalidade inerente ao *corpus*[2] a ser processado, pois na grande maioria, este *corpus* apresenta centenas de documentos fazendo com que o número de características ou termos possam ultrapassar a centena de milhares com muita facilidade gerando um problema muito conhecido em PLN denominado da maldição da dimensionalidade.

No entanto, a abordagem BOW tem apresentado bons resultados na literatura, justificando assim a sua abordagem puramente estatística. BOULIS *et al*. (2005) afirmaram que apesar da simplicidade da representação BOW, os métodos de classificação que fazem uso da mesma, muitas vezes apresentam um alto desempenho. TURNEY *et al*. (2010) também reportaram que os motores de busca que fazem uso desta hipótese funcionam surpreendentemente bem e que a mesma parece capturar algum aspecto importante da semântica das palavras.

4.1.1. Representação Distribuída das palavras

Na seção anterior foi apresentada uma forma de representar os documentos como vetores em um espaço vetorial, onde dois documentos similares tendem a ter palavras similares e consequentemente seus vetores são similares (SALTON *et al*., 1975). Entretanto a representação vetorial também pode ser utilizada para representar o significado das palavras, associando cada palavra a um vetor, onde os valores deste vetor estão relacionados a distribuição das palavras que são vizinhas a palavra analisada.

Diferentemente da hipótese BOW citada anteriormente este tipo de representação está associada a hipótese da distribuição, formulada pela primeira vez por (HARRIS, 1954), onde o mesmo afirmou que as palavras que ocorrem em contextos similares, ou seja, com os

[2] Corpus (plural de corpora), qualquer coleção de textos ou base de dados textuais.

mesmos vizinhos, tendem a ter significados similares. Entretanto, a declaração mais famosa desta representação foi dada por um linguista em 1961 (RUPERT, 1961), onde o mesmo afirmou que é possível conhecer uma palavra através da companhia que ela mantém (do inglês: "*You shall know a word by the company it keeps*!").

Dentro deste contexto, citando como exemplo a palavra "escada" é possível afirmar então que em um texto, existe uma grande probabilidade da mesma, aparecer ao lado de palavras como, "subir", "descer", "lâmpada", "construção", "piso", etc. Estas palavras representam termos que co-ocorrem com a palavra citada (escada), ou seja, são seus possíveis vizinhos. Sendo assim é possível afirmar que na hipótese distribuída procura-se todas as ocorrências de uma determinada palavra e usando uma janela de tamanho *k* pré-definido em torno da palavra analisada é possível encontrar seus vizinhos (todas as palavras dentro desta janela serão consideradas como vizinhas da palavra analisada). Na figura 4.3 apresentada a seguir é possível visualizar os vizinhos (contexto) de uma sentença definida como: "Ele subiu na escada para trocar uma lâmpada queimada". O contexto da palavra analisada (destacada em verde) é definido a partir de uma janela (*k*) de tamanho igual a 2.

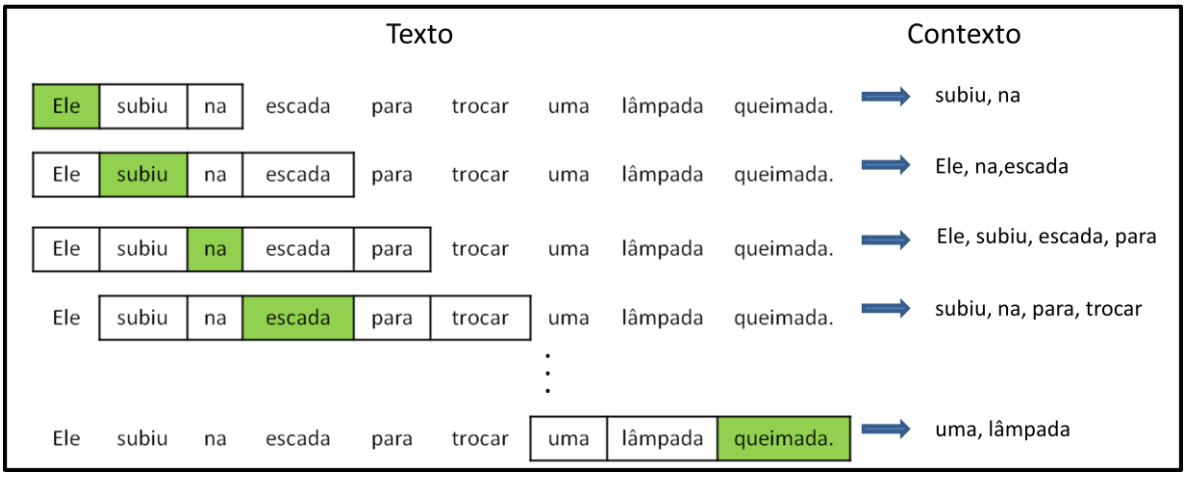

Figura 4.3 - Contexto das palavras utilizando representação distribuída

O princípio da hipótese distribuída pode ser analisado em diferentes abordagens que estão divididas em duas categorias diferentes (BARONI *et al.*, 2014) conforme ilustrado na figura 4.4. Na primeira categoria estão os métodos baseados em contagem, que faz uso da estatística para determinar com que frequência as palavras co-ocorrem. Neste caso é construído uma matriz de co-ocorrência das palavras para um determinado conjunto de treinamento (*corpus*), onde cada linha representa uma palavra no *corpus* e cada coluna é o contexto em que a palavra foi observada dentro de uma determinada janela em torno da palavra analisada (esta busca pelo contexto da palavra pode ser visualizada na figura 4.3

apresentada anteriormente). Geralmente o cálculo da frequência com que as palavras co-ocorrem pode ser determinado pela medida de Informação Mútua Pontual (*Pointwise Mutual Information* - PMI) proposta por (CHURCH *et al*., 1989), que é baseada na noção de informação mútua (para mais detalhes sobre esta e outras medidas veja seção 6.2.1). JURAFSKY *et al*. (2014) afirmaram que a PMI é mais eficiente como medida da associação entre duas palavras do que o simples cálculo da frequência.

Uma abordagem muito utilizada da hipótese distribuída baseada em contagem é a Análise Semântica Latente (*Latent Semantic Analysis* - LSA) proposta por (DEERWESTER *et al*. 1990). Nesta abordagem após a criação da matriz de co-ocorrência das palavras é feito uma redução da dimensionalidade desta matriz utilizando o método de Decomposição em Valores Singulares (*Singular Value Decomposition – SVD*), (GOLUB *et al*., 1996). Esta redução dimensional permite uma diminuição dos ruídos presentes na amostra gerando vetores mais densos (DEERWESTER *et al*. 1990). O cálculo da similaridade entre vetores pode ser realizado através da medida do cosseno apresentada na equação 4.3. Trabalhos envolvendo a LSA podem ser encontrados em (EVANGELOPOULOS *et al*., 2012; YALCINKAYA *et al*., 2015).

Na segunda categoria estão os métodos preditivos que tentam diretamente prever uma palavra a partir de seus vizinhos utilizando algoritmos de aprendizagem de máquina. Os vetores gerados são denominados *word embeddings* e são considerados como parâmetros do modelo. Exemplos de métodos preditivos são o Modelo de linguagem Neural (*Neural Network Language Model – NNLM*) (BENGIO *et al*., 2003), o método *Global Vectors for Word Representation - Glove* (PENNINGTON *et al*., 2014) e o *Word2vec*, nas versões *Skip-Gram* e *Cbow* (MIKOLOV *et al*., 2013). Sendo este último detalhado na próxima seção, pois o mesmo foi utilizado nesta tese na etapa de representação das palavras.

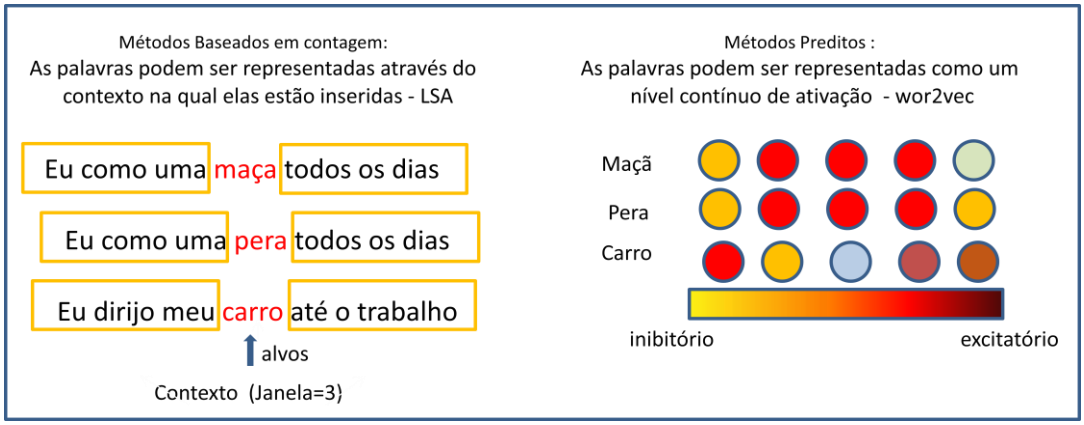

Figura 4.4 - Representação distribuída (à esquerda, métodos baseados em contagem e à direita métodos preditivos)

4.2. Word2Vec

Os métodos preditivos da hipótese distribucional vêm sendo muito citados na literatura nos últimos anos. Este sucesso se deu devido à publicação do trabalho de (MIKOLOV *et al*., 2013a), onde os autores implementaram um algoritmo denominado *word2vec* que consiste em criar representações vetoriais de palavras (*word embeddings*) utilizando técnicas de RNAs. Ou seja, ao invés de contar co-ocorrências de palavras vizinhas (como nos métodos baseados em contagem), o vetor é predito através de métodos de aprendizado de máquina capturando assim indiretamente propriedades linguísticas para modelar a linguagem.

Estes vetores *word embeddings* vem sendo utilizados em diferentes aplicações de PLN como em classificação de textos, agrupamento de documentos, rotulador de categorias morfossintáticas (*part of speech tagging*), reconhecimento de entidade nomeada (*named entity recognition*), análise de sentimentos, entre outras.

A terminologia *word embedding* vem da comunidade de DLNN, porém é possível encontrar na literatura o termo modelo semântico distribuído (*distributional semantic model*), representação distribuída, vetor de espaço semântico (*semantic vestor space*) ou simplesmente espaço de palavra (*word space*). Neste trabalho aderindo a prática atual em aplicações de DLNN foi adotado o termo *word embedding*.

O uso de *word embedding* foi proposto pela primeira vez por (BENGIO *et al*., 2003) no modelo denominado modelo de linguagem de redes neurais (*Neural Network Language Model - NNLM*) que teve como base as ideias de representação distribuída proposta por (HINTON, 1986). Entretanto o uso da mesma só se tornou popular com o algoritmo *word2vec* proposto por (MIKOLOV *et al*., 2013a).

O algoritmo *word2vec* é um modelo preditivo que visa aprender os vetores *word embeddings* através de um modelo de RNA. E apesar de ser considerado como um grande motivador do uso de DLNN em PLN, o mesmo não faz uso das técnicas de aprendizado profundo e sim de RNAs de arquitetura rasa (*shallow*). Pode se dizer então que esta importância está no fato de que a saída do *word2vec* é que pode ser entendida e utilizada facilmente pelos modelos de redes DLNNs.

O objetivo do *word2vec* é agrupar palavras semanticamente semelhantes no espaço vetorial, ou seja, o algoritmo consegue criar uma representação distribuída das palavras de forma eficiente preservando as relações semânticas que as palavras têm com seus vizinhos. A

forma como isto pode ser feito é através de operações simples entre vetores. Em um trabalho proposto por (MIKOLOV *et al*., 2013b), os autores mostraram que ao realizar a operação v(*king*) – v(*man*) + v(*women*) o vetor palavra mais próximo foi v(*queen*), que representa a figura feminina do rei (v(*king*)). Vale ressaltar que este resultado inicialmente foi considerado como notável, uma vez que para geração de modelos usando o *word2vec* é necessário apenas um conjunto de textos, que podem ter sido pré-processados ou não. Nenhum conhecimento sintático ou semântico precisa ser passado ao algoritmo, pois o mesmo captura estas relações através da análise da posição das palavras e do contexto que as acompanha.

Existem dois principais algoritmos de aprendizado para o *word2vec*, o *Continuous-Baf-of-Words* (CBOW) e *Continuous Skip-Gram*. Algoritmicamente os dois modelos são similares, entretanto no CBOW o algoritmo prevê a palavra corrente (alvo) baseada no contexto (uma janela de palavras próximas da palavra alvo) enquanto o *Continuous Skip-Gram* faz o inverso, prevê o contexto baseado na palavra corrente.

Esta inversão, estatisticamente tem o efeito de que o *skip-gram* representa melhor as palavras quando o conjunto de dados é pequeno e praticamente é mais utilizado pois na maioria das vezes as aplicações não disponibilizam de grandes bancos de dados. Já o CBOW, entretanto é mais rápido e mais adequado para grandes bancos de dados (MIKOLOV *et al*., 2013).

O modelo *Skip-gram* foi escolhido como método para representação das palavras nesta tese. Esta escolha está em linha com a proposta de (MIKOLOV *et al*., 2013a) citada anteriormente, onde os autores afirmam que o método *Skip-gram* é o mais adequado para conjunto de dados pequenos, que é o caso da base de dados utilizada.

4.2.1. Skip-gram

O modelo *Skip-gram*, conforme mencionado anteriormente, prevê o contexto baseado na palavra corrente, ou seja, na camada de saída da rede neural é realizada a previsão do contexto. Em outras palavras, pode se dizer que o *Skip-gram* é basicamente um modelo de rede neural com uma camada escondida. O mesmo pode ser visualizado na figura 4.5 apresentada a seguir. Nesta figura, V é o tamanho do vocabulário, N é a quantidade de neurônios da camada escondida, x e y são vetores *one-hot* (para detalhes veja início desta seção) que representam as entradas e saídas do modelo respectivamente, W_{VxN} é a matriz de pesos entre a camada de entrada e a camada escondida, onde a *i*-esima linha da matriz

representa o peso correspondente da *i*-esima palavra do vocabulário, a matriz W'_{VxN} que representa os pesos entre a camada escondida e a camada de saída e C que está relacionado ao tamanho do contexto (quantidade de vizinhos da palavra alvo).

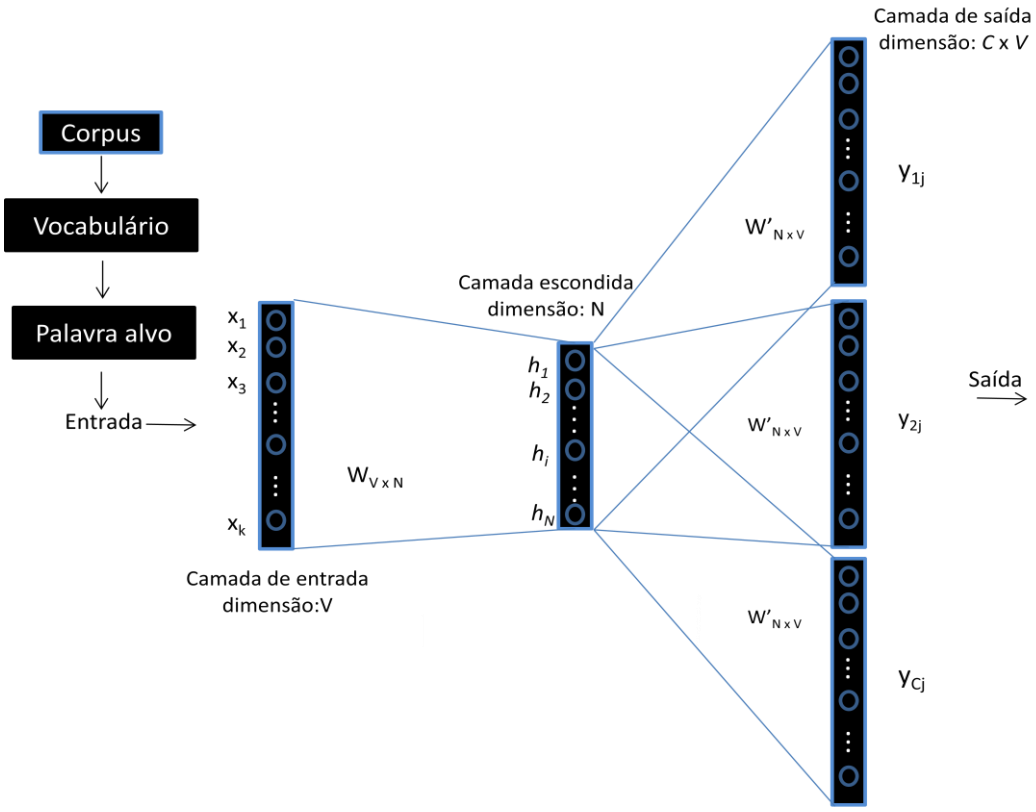

Figura 4.5 - Modelo *Skip-Gram*. Adaptado de (RONG, 2014)

Sendo assim, dado uma sequência de palavras do conjunto de treinamento (w_1, w_2, w_3,.....,w_T), o objetivo do algoritmo é maximizar a média do logaritmo da probabilidade conforme a equação 4.4, onde *C* é o tamanho da janela de treinamento (possíveis vizinhos da palavra w_t).

$$\frac{1}{T}\sum_{t=1}^{T}\left[\sum_{j=-C,j\neq 0}^{C}\log p(w_{t+j}\mid w_t)\right] \tag{4.4}$$

Toda palavra *w* no modelo *skip-gram* está associada com dois vetores de treinamento (u_w e v_w) que são aprendidos ao longo do treinamento. Estes vetores estão associados aos vetores de entrada e de saída da palavra *w* respectivamente. A probabilidade da predição correta da palavra w_i dada a palavra w_j pode ser definida pelo modelo *softmax* apresentado pela equação 4.5, onde V é a quantidade de palavras no vocabulário.

$$p(w_i \mid w_j) = \frac{\exp(u_{w_i}^T v_{w_j})}{\sum_{l=1}^{V} \exp(u_l^T v_{w_j})} \tag{4.5}$$

O treinamento do modelo geralmente é realizado utilizando o gradiente descendente estocástico com algoritmo *backpropagation* proposto por (RUMELHART *et al*.,1986) e detalhado no capítulo 2 desta tese. O uso do modelo *skip-gram* com o *softmax* é uma operação que tem um alto custo computacional, porque processar log ($w_i|w_j$) é proporcional a quantidade de palavras presentes no vocabulário (V) e este número pode chegar na ordem de trilhões de palavras muito facilmente o que torna esta operação muito custosa. MIKOLOV *et al*., 2013) afirmaram que para tornar o algoritmo mais eficiente e reduzir a complexidade do modelo uma alternativa é fazer uso do *softmax* hierárquico proposto por (MORIN *et al*., 2005) ou através do método de amostragem negativa (*negative sampling*), ambos detalhados nas próximas seções.

4.2.1.1. Softmax Hieráquico

Uma otimização computacional eficiente para o uso do modelo *skip-gram* pode ser realizada através de técnicas de aproximação para o cálculo das probabilidades de seleção de cada uma das palavras. Uma das técnicas citadas na literatura e que vem sendo utilizado com sucesso é a abordagem do *softmax* hierárquico. A principal vantagem deste método é que para um determinado conjunto de treinamento de tamanho V ao invés de se avaliar V nós de saídas para obter a distribuição de probabilidade, é necessário avaliar apenas $\log_2(V)$ nós, limitando assim o número de vetores saídas que devem ser atualizados no treinamento.

Este modelo hierárquico faz uso de uma representação de árvore binária para representar todas as palavras do vocabulário (V). Cada palavra em V representa uma unidade das folhas da árvore e para cada unidade folha, existe um caminho único que vai desde de a raiz até a unidade. E este caminho é utilizado para estimar a probabilidade das palavras representadas pelas unidades folha.

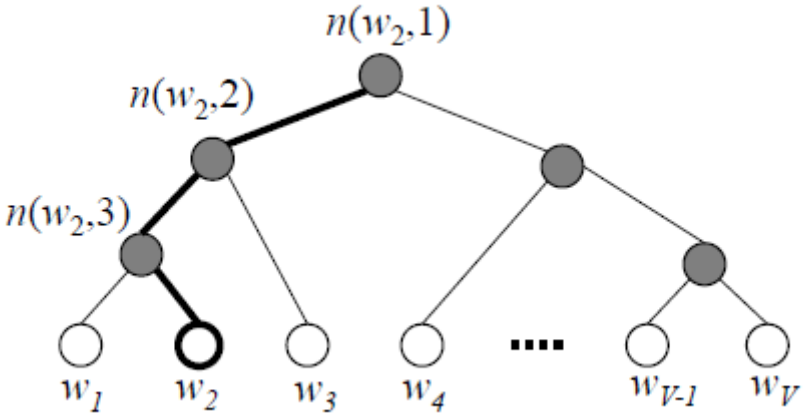

Figura 4.6 - Exemplo de árvore binária utilizando *softmax* hierárquico (RONG, 2014)

Na figura 4.6 os nós brancos representam as palavras no vocabulário (V) e os nós escuros são os nós internos. Um exemplo de caminho da raiz da árvores até a palavra *w2* é destacado tendo como tamanho 4, ou seja, L(*w2*) = 4. A expressão *n(w,j)* indica o *j-ésimo* nó no caminho da raiz até a palavra *w*.

No modelo *softmax* hierárquico cada nó intermediário V-1 tem um vetor saída $v'_{n(w,j)}$ e a probabilidade de uma palavra ser considerada como saída do modelo é dada pela seguinte equação:

$$p(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma([[n(w, j+1) = ch(n(w, j))]].v'^{T}_{n(w,j)} h) \tag{4.6}$$

Onde *ch(n)* é o nó filho à esquerda do nó *n*; $v'_{n(w,j)}$ é a representação ("vetor de saída") do nó intermediário *n(w,j)*, h é a saída da camada escondida, $\sigma(x)$ é a função sigmoide e [[x]] é uma função especial definida em 1 quando x é verdadeiro e -1 caso contrário. Sendo assim, para se calcular intuitivamente qualquer palavra de saída é necessário ter a probabilidade de cada nó intermediário existente no caminho entre a raiz e o nó da palavra de saída. Esta probabilidade pode ser atribuída indo para a esquerda ou para a direita em cada nó intermediário. A probabilidade de ir para a esquerda em um nó intermediário (n) é dada por:

$$p(n, left) = \sigma(v'^{T}_{n}.h) \tag{4.7}$$

E a probabilidade de ir para o nó da direita é:

$$p(n, rigth) = 1 - \sigma(v'^{T}_{n}.h) = \sigma(-v'^{T}_{n}.h) \tag{4.8}$$

Desta forma, considerando a figura 4.6 apresentada anteriormente, a probabilidade de se classificar o nó w2 como saída pode ser dada por:

$$p(w_2 = w_O) = p(n(w_2,1), left).p(n(w_2,2), left).p(n(w_2,3), right) \quad (4.9)$$

$$= \sigma(v'^{T}_{n(w_2,1)}.h).\sigma(v'^{T}_{n(w_2,2)}.h).\sigma(-v'^{T}_{n(w_2,3)}h) \quad (4.10)$$

Este resultado corresponde exatamente ao mesmo resultado apresentado na equação 4.6. E também é possível verificar através da equação 4.6 que:

$$\sum_{i=1}^{V} p(w_i = w_O) = 1 \quad (4.11)$$

Segundo (MIKOLOV *et al*., 2013) é possível afirmar então que o *softmax* hierárquico é uma distribuição multinomial bem definida entre todas as palavras e que isto implica que o custo computacional de se calcular a função de perda e seu gradiente é proporcional ao número de nós (V) existentes no caminho intermediário entre o nó raiz e o nó de saída, que em média não é maior que o log (V).

4.2.1.2. Método de Amostragem Negativa

Uma outra abordagem que visa a redução do custo computacional no modelo *skip-gram* e diferente da alternativa citada anteriormente (*Softmax* hierárquico) é o método de amostragem negativa proposto por (MIKOLOV *et al*., 2013). Neste trabalho os autores fizeram uma proposta de simplificação do modelo *Noise Contrastive Estimation* (NCE) introduzido por (GUTMANN *et al*., 2012) e utilizado por (MNIH *et al*., 2012) em um modelo de linguagem.

O modelo NCE afirma que um bom modelo deve ser capaz de diferenciar dados de ruídos através da regressão logística. O objetivo no NCE é aprender parâmetros para um modelo arbitrário de linguagem, enquanto o modelo de amostragem negativa visa a geração de representação distribuída de palavras de maneira eficiente ao invés de modelar a distribuição de palavras em linguagem natural.

Neste caso, o modelo é treinado para discriminar a palavra correta de *N* palavras ruidosas levando em consideração um determinado contexto e sendo definida através da seguinte equação:

$$\log\sigma(v'^{T}_{w_O} v_{w_I}) + \sum_{i=1}^{N} E_{w_i} \sim P_n(w)\left[\log\sigma(-v'^{T}_{w_i} v_{w_I})\right] \tag{4.12}$$

Onde w_o é a palavra de saída (amostra positiva), v'_{w_O} é o vetor de saída, v_{w_I} é o vetor de saída da camada intermediária, $P_n(w)$ é a distribuição de probabilidade de ruído (amostra negativas) e N é a quantidade de palavras negativas amostradas para cada contexto, v'_{w_i} é o vetor de saída da amostra negativa e $\sigma(x) = \frac{1}{1+e^{-x}}$.

Em (MIKOLOV *et al.*, 2013) os autores afirmaram que a distribuição de ruído $P_n(w)$ é um parâmetro livre no modelo, mas no entanto a distribuição Unigrama U(w) definida e apresentada a seguir foi a distribuição com melhores resultados nos experimentos realizados pelos autores.

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{N} f(w_j)^{3/4}} \tag{4.13}$$

O valor de ¾ citado na equação acima foi o valor encontrado pelos autores nestes experimentos e *f(w)* é a frequência da palavra no *corpus* analisado.

4.2.2. Vantagens do Word2Vec

Dentro do contexto de PLN (talvez uma das áreas mais desafiadoras de *Machine Learning* no atual momento da elaboração desta tese) o uso da representação distribuída de palavras através do algoritmo *word2vec* vem ganhando cada vez mais espaço na literatura e na tomada de decisão de suas diferentes áreas de aplicações. O grande sucesso do *word2vec* e sua popularidade podem ser atribuídos principalmente à distribuição de código aberto do pacote de software proposto por (MIKOLOV *et al.*, 2013) com códigos que podem facilmente ser acessados ou mesmo devido a disponibilidade de *word embeddings* já pré-

treinadas que podem ser utilizadas diretamente como características (*features*) em diferentes aplicações de PLN.

SCHNABEL *et al.* (2015) afirmaram que o *word2vec* é capaz de aprender representações vetoriais de alta qualidade e baixa dimensionalidade (geralmente 50 - 200) e que o mesmo ainda consegue capturar automaticamente padrões linguísticos presentes nos dados e dentro do espaço vetorial codificado. Os autores afirmaram ainda que além de conseguir processar grandes bancos de dados com eficiência, o *wor2vec* também é eficiente em representações semânticas devido à sua compactação e geometria capturando assim informações de inter-relação entre as palavras.

A facilidade na geração de modelos usando o *word2vec* é outra vantagem que é muito destacada na literatura, ou seja, para o algoritmo não há necessidade de se passar nenhum conhecimento sintático ou semântico do texto a ser analisado, pois o mesmo captura estas relações através da análise da posição das palavras e do contexto das palavras analisadas. O texto analisado não precisa ser pré-processado. A lista de vetores multidimensionais resultantes do processamento do algoritmo e que representam as palavras no texto, podem ser facilmente analisadas através de um dicionário distribucional. E a comparação de cada palavra para todas as outras palavras neste dicionário podem ser realizadas facilmente através do cálculo da distância (do cosseno). Sendo assim, para um determinado texto analisado onde o dicionário resultante é uma lista no formato: São Francisco e Los Angeles = 0.6661, pode-se afirmar então que a distância dos cossenos entre o vetor que representa São Francisco e o vetor Los Angeles no modelo é de 0.6661. Na figura 4.7 é possível visualizar a distribuição destes vetores geometricamente em um espaço bidimensional através da técnica de Análise de Componentes Principais (*Principal Component Analysis* – PCA). Na figura também é possível verificar como o algoritmo consegue identificar com sucesso diferentes países e suas respectivas capitais, comprovando assim a habilidade do modelo de organizar automaticamente conceitos e aprender implicitamente relações entre os mesmos.

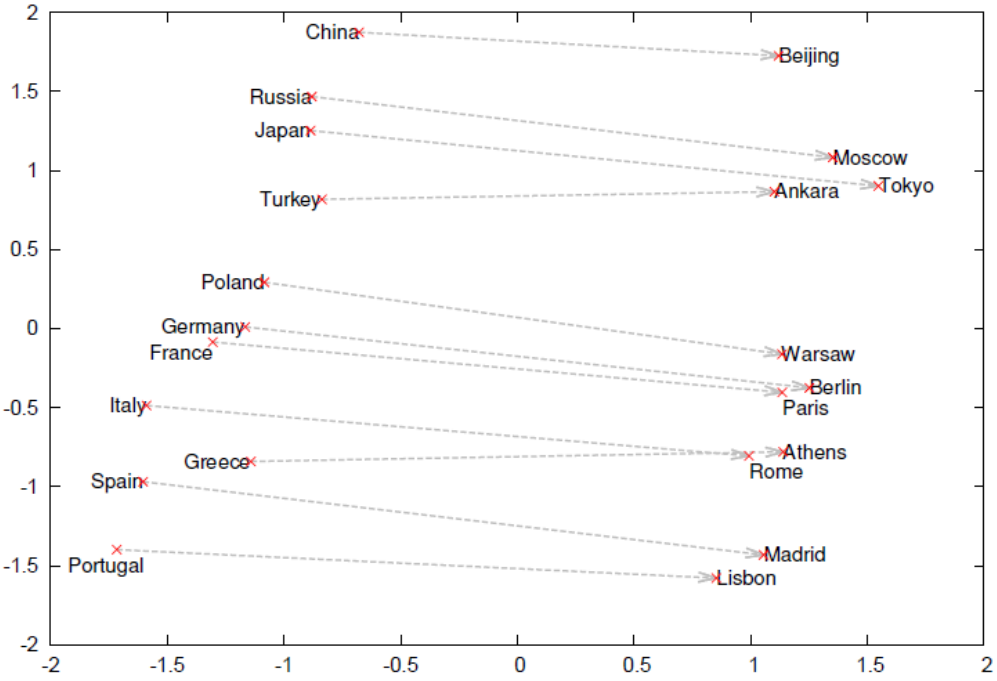

Figura 4.7 - Projeção PCA de vetores *word2vec Skip-gram* de diferentes países e suas respectivas capitais (MIKOLOV *et al*., 2013)

Por fim cabe ressaltar que a escolha pela utilização do *word2vec* modelo *skip-gram* e consequentemente pela representação distribuída (*word embeddding*) no trabalho proposto se deve a todas as características mencionadas anteriormente e também a resultados recentes reportados na literatura, que mostram uma superioridade do mesmo às abordagens existentes até o momento (BARONI *et al*., 2014).

5. Trabalhos Correlatos

As redes Neurais DLNNs recentemente se tornaram ferramentas poderosas para resolver problemas complexos devido principalmente às melhorias nos seus algoritmos de treinamento. Exemplos de aplicações bem-sucedidas do uso de DLNN podem ser encontradas em reconhecimento de fala e caracteres, visão computacional e reconhecimento facial. Entretanto, o uso da mesma em aplicações financeiras ainda é tímido e com resultados poucos expressivos devido à alta complexidade envolvendo as diversas variáveis que regem os diferentes mercados financeiros mundiais.

A abordagem de dados que permite o uso de DLNN em mercados financeiros pode ser através de dados estruturados (históricos de preços das ações, indicadores técnicos, etc.) ou dados não estruturados (notícias financeiras, relatos financeiros de empresas, entre outros), como é o caso desta tese. Dentro do contexto de dados estruturados e DLNN alguns trabalhos promissores foram publicados recentemente, (BAO *et al*., 2017; KORCZAK *et al*., 2017 e NAVON *et al*., 2017). Estes três trabalhos apresentaram uma estrutura de aprendizado profundo visando uma melhoria nas previsões de séries temporais financeiras e os resultados reportados pelos autores provaram a eficiência e a robustez destas estruturas nos modelos desenvolvidos.

Já a utilização de modelos de previsão do mercado financeiro por dados não estruturados ainda é uma área recente de pesquisa, mas a mesma vem ganhando popularidade atualmente principalmente com o avanço de técnicas de PLN (como por exemplo: os recentes algoritmos para representação distribuída de palavras: *word2vec* e *Glove*); o surgimento das redes DLNNs e o crescimento exponencial de dados não estruturados (relatórios financeiros, comunicados à imprensa, artigos de notícias financeiras, artigos de sites de mídia social como o *Twitter*, entre outros). De fato, conforme citado na introdução desta tese, no trabalho proposto por (DING *et al*., 2015), os autores demonstraram uma aplicação bem-sucedida e promissora do uso das redes DLNNs juntamente com a representação distribuída de palavras na previsão do retorno diário do S&P500.

O uso de redes neurais DLNNs e *word embedding* visando a previsão do mercado de ações também foi proposto por (PENG *et al*., 2015). Neste trabalho os autores utilizaram dados fornecidos por (DING *et al*., 2015) nos quais contém 106521 artigos publicados pela empresa *Reuters* e 447145 da *Bloomberg* no período de outubro de 2006 a dezembro de 2013.

Os preços históricos das ações foram fornecidos pelo *Centre for Research in Security Prices*[3] (CRSP). Os dados foram divididos em 3 subconjuntos (treinamento, validação e teste). O objetivo principal do trabalho foi o uso de DLNN como modelo para prever o preço de subida e descida das ações no futuro, onde a mesma recebe como entrada um conjunto de características extraídas a partir de informações históricas de preços das ações e notícias financeiras disponíveis na internet. A estrutura da DLNN utilizada neste trabalho correspondeu a uma rede neural MLP com 4 camadas ocultas (sendo 1024 neurônios em cada camada), função de ativação ReLu nas camadas escondidas e para a camada de saída foi utilizada a função *Softmax* responsável por calcular a probabilidade a posteriori entre dois nós da rede, definindo assim a subida ou descida do preço da ação.

Este estudo fez uso de três tipos de características que foram definidas a partir do histórico dos preços das ações; dos resultados obtidos a partir do processamento das notícias financeiras e do uso do algoritmo *word2vec*. Em relação ao histórico dos preços das ações, os autores escolheram e concatenaram uma janela de 5 dias (t-5, t-4, t-3, t-2 e t-1) anteriores ao dia de previsão analisado (t) gerando um vetor característica (P) que foi utilizado como entrada para a rede DLNN. Os dados foram normalizados pela média e variância de todo o conjunto de dados de treinamento da DLNN. Ademais, foram concatenadas também a este vetor característica (P) as diferenças entre a primeira e a segunda ordem (ΔP e $\Delta\Delta P$ respectivamente) entre os preços de fechamento dos cinco dias analisados na janela. Sendo assim, para cada ação alvo a ser prevista, o vetor de características que representa o preço histórico das ações foi composto de $P + \Delta P + \Delta\Delta P$ características. Já em relação ao processamento das notícias financeiras, todos os artigos utilizados foram divididos em sentenças, onde foram mantidas apenas frases que continham pelo menos o nome de uma empresa ou ação. Cada sentença apresentava também além do nome da ação contida na mesma, a data de publicação da notícia original. Por fim, todas as sentenças foram agrupadas pela data de publicação das mesmas e o nome das ações formando assim o conjunto de dados de notícias. Cada amostra do conjunto de dados foi constituída de uma lista de frases que foi publicada na mesma data e que tenha mencionado a mesma empresa ou ação. Além disto, cada amostra foi também classificada como positiva ou negativa com base no preço de fechamento do dia seguinte adquirido através do banco de dados financeiros da CRSP. Uma vez que a base de dados de notícias foi composta e classificada, os próximos passos dos autores foram extrair três tipos de características para cada amostra. Para esta etapa, os

[3] CRSP é um provedor de dados históricos do mercado de ações e faz parte da *Booth School of Business* da Universidade de Chicago. Mantém alguns dos maiores e mais completos bancos de dados históricos proprietários em pesquisa de mercado de ações (http://www.crsp.com/).

autores fizeram uso do algoritmo *word2vec* para gerar características denominadas *Bag-of-Keywords* (BoK) e *Category tag* (CT) e da Informação Mútua Pontual (*Point-wise Mutual Information*) para gerar características de pontuações de polaridade (*Polarity score*). Conforme citado até o momento, os autores neste trabalho só fizeram uso de ações que foram mencionadas nas notícias utilizadas. Sendo assim, somente estas ações puderam ser previstas para a data analisada. A fim de tentar prever um outro grupo de ações, ou seja, aquelas que não estão diretamente mencionadas nas publicações diárias de notícias financeiras, os autores implementaram também um método de previsão destas ações utilizando um gráfico de correlações de ações, onde cada nó do gráfico representa uma ação e cada arco que conecta dois nós representa a correlação entre as duas ações representadas nestes nós. Sendo assim, a partir das previsões obtidas de ações mencionadas nas notícias e a metodologia citada anteriormente, os autores propagaram estes resultados ao longo dos arcos no gráfico para gerar correlações e possíveis previsões para ações não identificadas nas notícias analisadas.

Os resultados reportados pelos autores como melhor desempenho foi uma taxa de erro de 43,13% para o conjunto de teste que continham ações citadas na notícias e acurácia de 52,44% para ações não mencionadas nas notícias e previstas através do gráfico de correlações. Por fim, os autores concluíram que os experimentos realizados demonstraram que as notícias financeiras podem ser úteis na previsão de ações e que os métodos propostos podem melhorar significativamente a precisão da previsão para um conjunto de dados financeiros padrão.

Um estudo recente visando a previsão do índice S&P500 foi proposto por (HUYNH *et al.*, 2017). Neste trabalho os autores introduziram um modelo de previsão denominado *Bidirectional Gated Recurrent Unit* (BGRU). Assim como no trabalho citado anteriormente, os autores também fizeram uso de *word embeddings* obtidos através do processamento de notícias financeiras; do histórico dos preços das ações; das redes neurais DLNNs e de dados coletados nos sites da *Reuters* e *Bloomberg*. Os autores argumentaram que o uso de DLNN tem algumas vantagens quando comparadas a métodos de previsão tradicionais, como por exemplo: (i) os dados de entrada podem ser melhor explorados para a obtenção de itens relevantes para a previsão das ações; (ii) a não linearidade e iterações complexas entre os dados de entrada são considerados e (iii) evita mais facilmente o *overfitting*. Os experimentos reportados pelos autores foram responsáveis em avaliar o impacto de notícias financeiras sobre o preço das ações em diferentes intervalos de tempo (1, 2, 5, 7 e 10 dias). Cada notícia utilizada foi classificada como positiva ou negativa dependendo do valor de abertura e fechamento das ações no período analisado. Os resultados obtidos demonstraram que o

modelo proposto atingiu quase 60% de acurácia na previsão do índice S&P500 e 65% na previsão de ações individuais das empresas *Google*, *Wal-Mart* e *Boeing*.

A título de comparação, na tabela 5.1 apresentada a seguir pode ser visualizado os resultados obtidos pelos três modelos mencionados anteriormente. Estes modelos também foram comparados em (HUYNH *et al*., 2017) devido à similaridade dos mesmos na definição de suas características como, por exemplo: o uso do mesmo conjunto de dados, a mesma técnica de previsão (DLNN) e o mesmo objetivo (previsão do índice S&P500).

Tabela 5.1 - Resultados finais dos modelos citados

Autor	Acurácia
Ding *et al*., 2015	55,21%
Peng *et al*., 2015	56,87%
Huynh *et al*., 2017	59,98%

YOSHIHARA *et al.* (2017) introduziram também um trabalho visando a previsão do mercado de ações baseada em uma rede neural DLNN. Porém diferentemente dos modelos anteriores onde o mercado analisado foi o dos Estados Unidos (S&P500), neste trabalho, o mercado analisado foi o Japonês, ou seja, foram analisadas ações de 10 empresas que compõe o índice *Nikkei* da bolsa de Valores de Tóquio. Este estudo utilizou a edição matinal dos jornais *Nikkei* publicados no período de 1999 a 2008 para coletar as notícias a serem processadas. A base de dados textual gerada foi dividida em três subconjuntos (treinamento, validação e teste). Os alvos a serem preditos foram o índice *Nikkei Stock Average* (NSA) e ações de 10 empresas *Nikkei* cujos os nomes apareceram com maior frequência nos dados de notícias analisados. O indicador de negociação frequentemente utilizado em análise técnica denominado *Moving Average Convergence Divergence* (MACD) foi aplicado na classificação das notícias do modelo. Para transformar os artigos de notícias em vetores de palavras, primeiramente foi determinado um vocabulário (conjunto de termos ou características) através da representação *bag-of-words* utilizando o MeCab[4] como analisador morfológico e testes estatísticos. Os dados processados e classificados foram utilizados com entrada dos modelos de DLNNs utilizados, ou seja, uma combinação dos modelos *Restricted*

[4] MeCab é uma biblioteca de código aberto voltada para a segmentação de texto escrito no idioma japonês. Foi originalmente desenvolvido pelo Instituto de Ciência e Tecnologia de Nara e é atualmente mantido por Taku Kudou. Disponível em: *https://sourceforge.net/projects/mecab/*

Boltzmann Machines (RBM), *Deep Belief Networks* (DBN) e *Recurrent Neural Networks* (RNN). A avaliação do modelo proposto foi realizada a partir de dois pontos de vistas diferentes: primeiramente foi analisado o desempenho da classificação das tendências dos preços das ações (acertos e erros das previsões) e em uma segunda abordagem foi implementado um sistema de simulação de compra e venda de ações baseado nos resultados obtidos. Uma comparação com os modelos de SVM e DBN também foi realizada neste estudo. Os autores reportaram que os resultados obtidos foram satisfatórios e que em média a abordagem proposta mostrou uma taxa de erro menor quando comparada ao SVM. Já em relação à comparação com a rede DBN, os autores só conseguiram resultados superiores quando modificaram o período analisado. Por fim, os resultados obtidos utilizando a estratégia de simulação do mercado apesar de apresentarem desempenho negativo, quando comparados ao modelo SVM, obtiveram consistentemente menos perdas.

Um sistema de negociação automatizado foi proposto por (DOS SANTOS *et al*., 2017), cujo objetivo foi prever tanto, mudanças nos preços das ações de empresas que fazem parte do índice da bolsa de Valores de Nova York (S&P500), quanto a variação do próprio índice. Os autores argumentaram que o modelo proposto está em linha com (DING *et al*., 2015) e que o mesmo pode ser dividido em duas partes. Sendo a primeira parte responsável em construir uma representação das entradas do modelo, ou seja, a definição do modelo de linguagem a nível de caracteres. Enquanto a segunda parte está relacionada a Rede Neural Recorrente (RNN), responsável pela previsão, ou seja, a rede recebe a entrada processada na etapa anterior e gera a previsão de subida ou descida dos preços das ações ou índice ao longo do prazo analisado. Para o modelo de linguagem neural foi utilizado uma camada denominada pelos autores de *character embedding* com 256 unidades seguida por uma única camada de processamento LSTM com 1024 nós. Os caracteres foram primeiramente codificados em *bytes* para simplificar o processamento e o treinamento dos mesmos. O modelo procura pelo caractere correspondente e então atualiza seus pesos e prevê a distribuição de probabilidade sobre o próximo *byte*. Por fim, os caracteres gerados (*character embbedings*) e os pesos finais da LSTM são salvos e usados para inicializar as primeiras camadas da RNN responsável pela classificação ou previsão. O treinamento do modelo foi realizado através do método do gradiente descendente estocástico. A rede RNN tem as mesmas duas camadas do modelo de linguagem neural e mais uma camada adicional totalmente conectada com 512 neurônios e função de ativação ReLu. Esta última camada é responsável em prever a probabilidade de subida ou descida do preço das ações. O modelo foi treinado por 50 épocas utilizando o algoritmo Adam (KINGMA *et al*., 2014). Os dados de

notícias financeiras utilizados foram coletados através da *Reuters* e *Bloomberg* no período de outubro de 2006 a dezembro de 2013 assim como em (DING *et al*., 2015; PENG *et al*., 2015; HUYNH *et al*., 2017) e o histórico dos preços das ações das empresas e valores do índice foram obtidos através da *Thomson Reuters Tick History*[5]. Os autores reportaram que apenas os títulos das notícias foram utilizados no modelo de previsão proposto, o que está em concordância com as conclusões obtidas por (RADINSKY *et al*., 2012; DING *et al*., 2015), onde os mesmos demonstraram que o uso dos títulos produzia melhores resultados quando comparados ao uso da notícia completa. Com os dados processados foram realizados dois testes: previsão a curto prazo (*intraday*) e previsão no final do dia de negociação (*interday*). Para a previsão *intraday* primeiramente foram filtradas somente as notícias que continham o nome das empresas pertencentes ao índice S&P500 e os experimentos foram conduzidos visando prever se o último preço da ação analisada após uma hora seria maior do que o primeiro preço depois da publicação da notícia, usando data e hora da publicação. Na previsão *interday*, foi utilizado uma abordagem semelhante a proposta por (DING *et al*., 2015; VARGAS *et al*., 2017), nos quais as notícias de uma mesma empresa para cada dia analisado foram concatenadas e os testes realizados tiveram como objetivo prever se o preço da ação da empresa analisada no dia seguinte (t + 1) aumentava ou diminuía quando comparado com o preço do dia anterior (t). Testes também foram feitos com o índice S&P500 na mesma janela de tempo para ambas as abordagens (*intraday* e *interday*). A acurácia encontrada pelos autores nos experimentos realizados foram as seguintes: para a previsão *intraday* das ações e para o índice S&P500 foram 59,86% e 61,68% respectivamente; já em relação a previsão *interday* foi 63,34% para a previsão de ações individuais e 64,74% na previsão do S&P500. É possível verificar que o modelo proposto funciona melhor para a previsão de ações individuais. Por fim, os autores concluíram que o uso de *word embedding* a nível de caracteres é promissora e competitiva com modelos mais complexos que fazem uso de indicadores técnicos e métodos de extração de eventos e notícias financeiras.

A partir dos trabalhos citados anteriormente é possível afirmar que este tema ainda está em aberto e que o número de modelos propostos para previsão do mercado financeiro utilizando dados não estruturados (dados textuais) não é comparável ao número de modelos propostos onde dados estruturados são utilizados para esta mesma finalidade. Como foi mencionado alguns pesquisadores têm sido bem sucedidos até certo ponto, na previsão do mercado financeiro utilizando dados não estruturados, entretanto é possível perceber que existe uma certa semelhança entre os mesmos, principalmente em se tratando do tipo de

[5] https://github.com/philipperemy/financial-newsdataset

dados utilizados (artigos de notícias disponíveis pela *Reuters* e *Bloomberg*) e do mercado estudado (Bolsa de Valores dos Estados Unidos e índice S&P500). Isto é consequência da dificuldade de se obter dados não estruturados confiáveis e disponíveis para o estudo dos diferentes mercados mundiais.

Por tudo o que foi dito anteriormente faz com que este tema de pesquisa mostre um grande interesse na comunidade científica mundial recebendo um número cada vez maior de publicações e congressos.

6.Construção do modelo de previsão

Como já abordado algumas vezes em diferentes pontos deste trabalho, a previsão do mercado financeiro ainda é uma tarefa difícil devido a presença de dados ruidosos, não estruturados e com alto grau de incertezas. Diferentes modelos foram propostos nos últimos anos, visando entender o comportamento de mercados financeiros do mundo inteiro e em especial os mercados de ações dos Estados Unidos (*S&P500* e *Nasdaq*). Com o surgimento das redes DLNNs e o avanço nas pesquisas nos algoritmos de representação de palavras existe uma maior expectativa de que os valores de previsão encontrados pelos modelos desenvolvidos atualmente consigam de alguma forma representar a realidade dos mercados analisados. Dentro deste contexto a abordagem utilizada para a finalidade desta tese (previsão diária da tendência do principal índice do mercado de ações brasileiro) é a implementação de uma ferramenta automática de classificação de notícias financeiras através da combinação das redes CNN e ELM juntamente do algoritmo *word2vec*. O modelo proposto denominado I-CNNELM pode ser visualizado na figura 6.1.

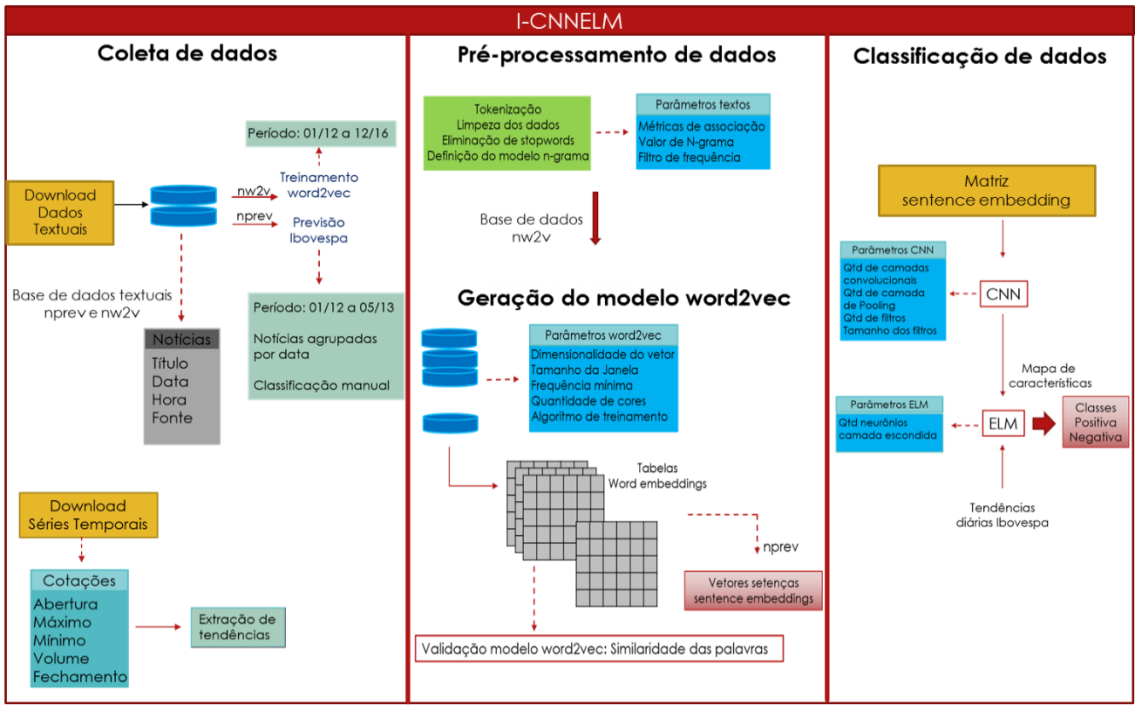

Figura 6.1 – Sistema de previsão proposto

De uma maneira geral, a metodologia proposta nesta tese é constituída de três etapas fundamentais: coleta de dados, a representação dos documentos (pré-processamento dos

dados e geração do modelo *word2vec*) e classificação das notícias. Por fim, o I-CNNELM foi avaliado a partir de diferentes métricas e uma estratégia de negociação foi implementada a fim de medir a lucratividade do mesmo. Para a implementação de todas as etapas citadas foi utilizado a linguagem de programação *Python* e algumas de suas bibliotecas como *Natural Language Toolkit* (NLTK), *Gensim*, *TensorFlow*, *Keras* e *Scikit-learn*. Algumas das características que serviram de motivação para o uso do *Python* e das ferramentas citadas são apresentadas a seguir:

- Permite uma máxima flexibilidade, velocidade e simplicidade na construção dos algoritmos;
- Apresenta um grande ecossistema de pacotes direcionados à comunidade de aprendizagem de máquina, visão computacional, processamento de sinais, processamento paralelo, imagem e vídeo, entre outros;
- São ferramentas multiplataformas, que pode processar uma grande quantidade de dados;
- Possibilita a implementação de topologias de redes neurais complexas com máxima flexibilidade, permitindo também a construção de gráficos arbitrários de redes neurais, assim como a paralelização das mesmas com o uso de mais CPUs e GPUs de uma forma eficiente.

Uma explicação de todas as etapas da metodologia proposta nesta tese, assim como mais detalhes das ferramentas citadas anteriormente são apresentados nas próximas seções.

6.1. Coleta de Dados

Para a realização do estudo proposto são necessários dois tipos de dados: (1) notícias financeiras contendo data e hora da publicação das mesmas e (2) séries temporais contendo as cotações diárias do fechamento do Ibovespa.

6.1.1. Base de dados de notícias financeiras

Os dados de notícias financeiras contendo data e hora de publicação foram adquiridas através dos principais sites de notícias econômicas (Infomoney, Exame, Estadão e G1) do

Brasil. Utilizando a linguagem de programação *Python* foi implementado um programa responsável por baixar automaticamente estas notícias da internet no período correspondente de 04 de janeiro de 2012 a 29 de dezembro de 2016, totalizando 380592 notícias. O programa implementado inicialmente faz o *download* destas notícias através de suas respectivas páginas *Html* e para cada página analisada, o programa separa o título, conteúdo da notícia, data e hora da publicação e o nome do veículo que publicou a notícia e armazena todas estas informações em arquivos textos (*.txt*). As demais informações contidas na página web são descartadas (como arquivos de cabeçalhos, rodapés, propagandas, *tags Html* de imagens, *links*, entre outros). O objetivo no final deste processamento é obter um texto sem formatação, imagens ou quaisquer outros efeitos que possam prejudicar as próximas etapas da metodologia proposta. A figura 6.2 apresenta as informações armazenadas pelo programa.

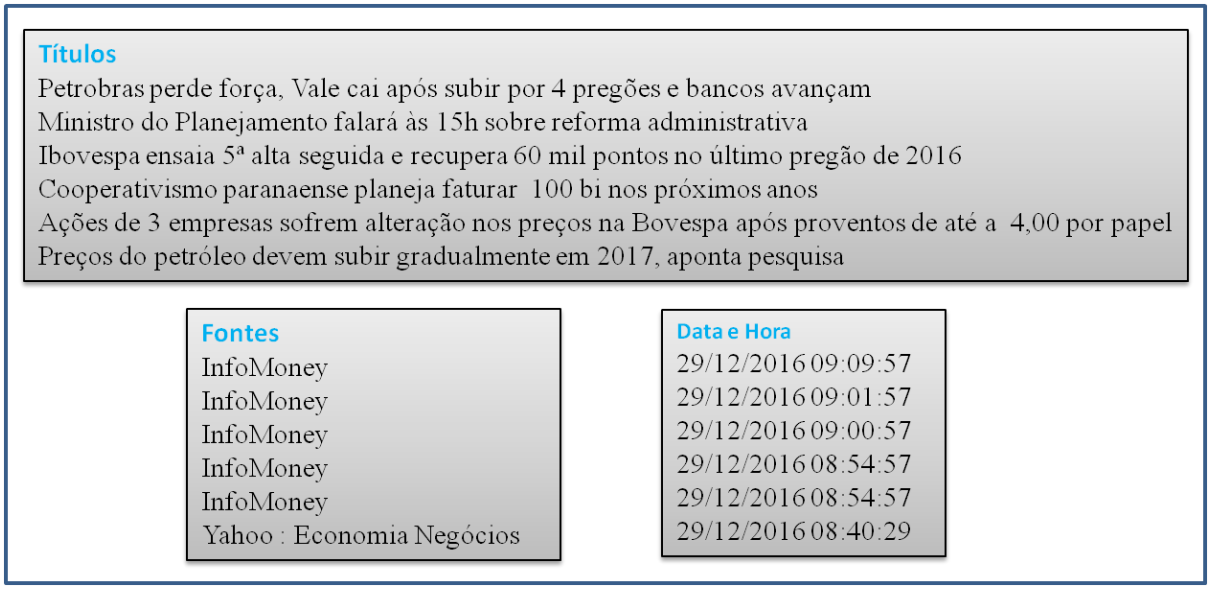

Figura 6.2 - Informações extraídas das páginas *web* analisadas

Conforme pode ser observado, a figura apresentada contém as seguintes informações: a data, horário, o site (ou jornal *on-line*) em que a notícia foi publicada e também o título da notícia contendo as informações financeiras (estas informações geralmente estão relacionadas a indicadores econômicos, situação econômica do Brasil e outros países como os Estados Unidos, Europa e Ásia, informações sobre mercado de ações internacionais, informações sobre taxas de juros, taxas de câmbio, etc.). Cada título da notícia armazenado está associado com a data e horário em que o mesmo foi publicado. Estas informações de data e hora são fundamentais para a definição do conjunto de dados textuais (títulos financeiros) e suas respectivas classes. A forma como é realizada a atribuição destas classes será detalhado na seção 6.1.3.

Uma vez que estas notícias foram armazenadas, o passo seguinte visou a definição de duas bases de dados a partir das notícias armazenadas, isto é, foi definida uma base de dados denominada nw2v para ser usada no treinamento do algoritmo *word2vec* e uma outra base de dados denominada nprev para ser usada no modelo de previsão do Ibovespa. Como o algoritmo *word2vec* necessita de uma grande quantidade de dados para ser treinado, a base de dados nw2v correspondeu às todas as notícias baixadas automaticamente pelo programa implementado, período de janeiro de 2012 a dezembro de 2016. Entretanto, para a base de dados nprev, a princípio a ideia foi analisar vários períodos separados (várias bases nprev) entre 2012 e 2016 e verificar como o modelo se comportaria diante de tantos problemas políticos e econômicos vivenciados no Brasil nestes últimos anos. Entretanto, com a demora na obtenção dos resultados e o fato das arquiteturas de redes sugeridas apresentarem um número grande de parâmetros a serem modificados, optou-se por utilizar apenas um período de tempo, porém variando um número maior de possíveis configurações de parâmetros. O período analisado correspondeu a janeiro de 2012 a maio de 2013.

Desta forma, dentro do contexto da base de dados nprev foi necessário agrupá-las por data. Sendo assim, todos os títulos de notícias pertencentes a um determinado dia de negociação, ou mesma data foram agrupados em um arquivo e separados por um ponto final (.), responsável por fazer a delimitação entre o fim de uma notícia e o começo de outra, conforme pode ser visualizado na figura 6.3.

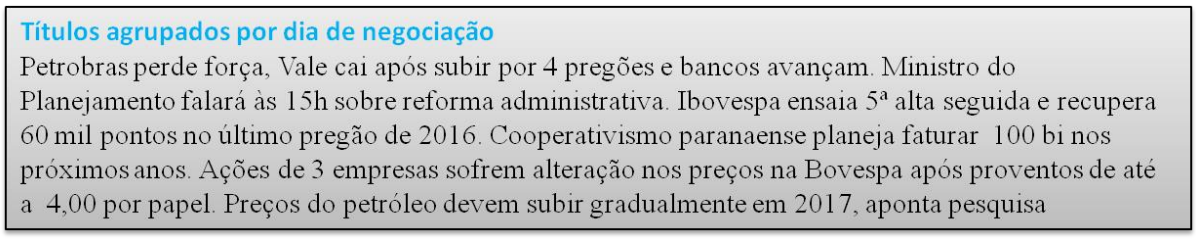
Títulos agrupados por dia de negociação
Petrobras perde força, Vale cai após subir por 4 pregões e bancos avançam. Ministro do Planejamento falará às 15h sobre reforma administrativa. Ibovespa ensaia 5ª alta seguida e recupera 60 mil pontos no último pregão de 2016. Cooperativismo paranaense planeja faturar 100 bi nos próximos anos. Ações de 3 empresas sofrem alteração nos preços na Bovespa após proventos de até a 4,00 por papel. Preços do petróleo devem subir gradualmente em 2017, aponta pesquisa

Figura 6.3 - Notícias agrupadas por data

A escolha pela utilização somente dos títulos das notícias e não de toda a notícia publicada está em linha com alguns dos trabalhos revisados no capítulo 5 desta tese, dentre os quais é possível citar (RADINSKY *et al.*, 2012; DING *et al.*, 2015). Nestes trabalhos os autores demonstraram que o uso dos títulos produzia melhores resultados quando comparados ao uso da notícia completa. Outra vantagem na utilização somente dos títulos de notícias está fundamentada no fato de que o título da notícia inclui um vocabulário bem definido contendo apenas a informação relevante, o que possibilita a criação de um conjunto de dados textuais

mais relevante e com menos ruído. Na tabela 6.1 podem ser visualizadas as estatísticas básicas da base nprev no período analisado.

Tabela 6.1 - Estatísticas básicas dos títulos das notícias no período analisado (janeiro/12 a maio/2013)

Total de Notícias (títulos)	**162001**
Mínimo de Notícias (títulos) diárias	**269**
Média de Notícias (títulos) diárias	**479**
Máximo de Notícias (títulos) diárias	**915**

6.1.2. Base de dados de séries temporais financeiras

Os dados referentes às séries temporais das cotações diárias do fechamento do Ibovespa foram adquiridos através de uma conceituada empresa de serviço de informações para o mercado financeiro. A série consiste de cotações diárias com valores de fechamento, abertura, máximo, mínimo (para detalhes ver capítulo 2) e volume negociado (que corresponde ao volume financeiro de transações realizadas em um determinado dia) no mesmo período relacionado a base de dados nprev (janeiro de 2012 a maio de 2013), totalizando 337 dias de negociação. Dentre os dados citados, apenas o valor diário do fechamento do Ibovespa foi utilizado e o mesmo pode ser visualizado na figura 6.4, onde é possível observar que o período analisado apresenta um comportamento típico do mercado financeiro (oscilações de altas e baixas dos preços) e de difícil predição.

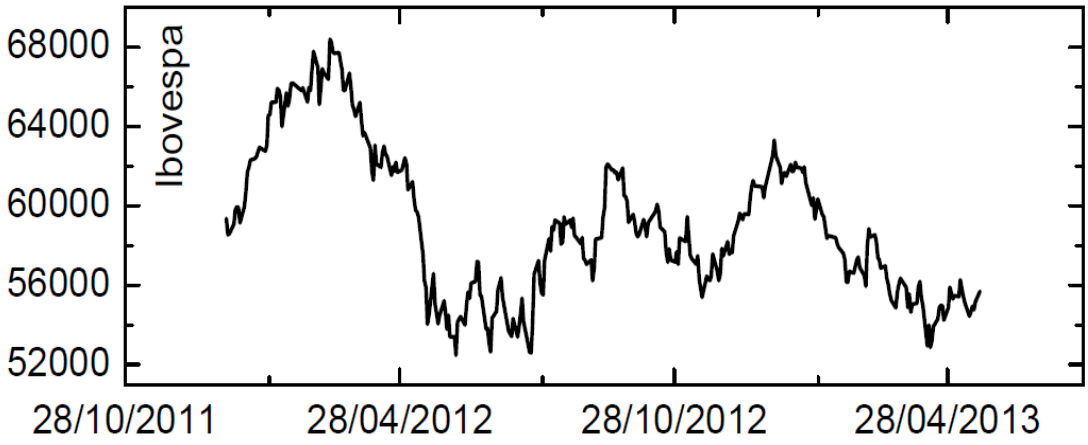

Figura 6.4 - Série diária do fechamento do Ibovespa (janeiro de 2012 a maio de 2013)

Do ponto de vista dos investidores de Bolsa de Valores as tendências das séries temporais financeiras são muito mais informativas do que o valor das ações (LAVRENKO *et al*., 2000; GIDÓFALVI *et al*., 2001) ou mesmo do próprio índice Bovespa. A extração de tendência em uma série temporal pode ser feita de maneira simples considerando apenas o valor de fechamento consecutivo das ações ou índices, ou mesmo através de algoritmos mais sofisticados, como a segmentação linear *piecewise* (KEOG *et al*., 2001). Neste trabalho em específico optou-se por extrair as tendências diárias do índice considerando o cálculo da variação relativa do Ibovespa, conforme a equação 6.1.

$$tendencia(t) = \frac{fechamento(t) - fechamento(t-1)}{fechamento(t-1)} * 100 \qquad (6.1)$$

Onde *t* é um dia de negociação do Ibovespa e *fechamento* está relacionado ao último valor assumido pelo Ibovespa no horário regular do pregão.

6.1.3. Classificação manual dos títulos de notícias

De uma maneira geral, a maioria dos trabalhos envolvendo previsão está relacionado a uma tarefa de PLN denominada classificação automática de textos. Na classificação de textos, técnicas de aprendizado de máquina, ou seja, um processo geral indutivo constrói automaticamente um classificador por "aprendizado", a partir de um conjunto de documentos

classificados previamente e características de uma ou mais categorias. Esta classificação prévia dos documentos ou processo de atribuição de classes geralmente é realizada por um especialista da área. Nesta tese, este processo é feito automaticamente baseado na hora em que as notícias foram publicadas e no valor da tendência diária do Ibovespa. Mais especificamente, a classificação foi realizada da seguinte forma: todas as notícias publicadas em um determinado dia de negociação da BmfBovespa são utilizadas para prever a tendência de seu fechamento no dia seguinte. Por exemplo: notícias publicadas no dia 05/08/2012 foram utilizadas para prever a tendência do fechamento do Ibovespa do dia seguinte (06/08/2012). Notícias publicadas em feriados e finais de semana são utilizadas para prever o a tendência do próximo dia de negociação do Ibovespa, ou seja, se as notícias foram publicadas no dia 02 de um determinado mês e este dia não é um dia de negociação da BmfBovespa, então estas notícias afetarão o mercado do próximo dia de negociação (03).

Para muitos autores na literatura, determinar o tempo (*timing*) que o mercado leva para reagir à publicação de notícias financeiras é fundamental para obter boas previsões. Esta tese segue os trabalhos propostos por (FEHRER *et al*., 2015; DING *et al*., 2015), onde os mesmos afirmaram que é possível fazer previsão dos preços das ações um dia a frente. Entretanto, outros autores afirmam que este *timing* pode ser menor como no trabalho proposto por (GIDÓVALFI, 2001), onde o autor reportou que existe uma forte correlação entre as notícias e o comportamento do preço das ações durante os 20 minutos que antecedem à publicação da notícia e os 20 minutos seguintes, logo o *timing* de previsão deve ser 20 minutos. Já (LAVRENKO *et al*., 2000) sugeriram um *timing* de cinco horas, ou seja, o mercado demora cinco horas para reagir à publicação de uma determinada notícia. O fato é que não existe atualmente uma fórmula matemática ou uma metodologia para determinar quando realmente acontece a incorporação das notícias nos preços das ações. O que é possível afirmar é que os investidores de uma maneira geral antes de tomar suas decisões leem cuidadosamente notícias econômicas e financeiras recentes para ter uma visão atualizada do mercado. Usando seus conhecimentos de como o mercado se comportou no passado em diferentes situações, estes investidores vão corresponder implicitamente à situação atual levando em consideração situações no passado que sejam semelhantes à atual. Dentro deste contexto, os modelos de previsão buscam através de algoritmos de aprendizagem reproduzir este conhecimento humano e prever situações em um período futuro, onde a determinação deste período de previsão ainda é motivo de muitos estudos e testes.

Além do *timing* existe também uma outra característica dos modelos de previsão que é a quantidade de classes a ser adotada para classificar os documentos (ou títulos de notícias). Muitos dos trabalhos envolvendo a previsão da tendência do mercado acionário citados na literatura fazem uso de modelos de três classes, ou seja, positiva, negativa e neutra, (LAVRENKO *et al*., 2000; GIDÓVALFI, 2001). Outros preferem adotar um modelo de apenas duas classes como em (ZHAI *et al*., 2007). Nesta tese optou-se pela utilização de duas classes, seguindo a dissertação de mestrado da própria autora (FARIA, 2012), onde o mesmo mercado acionário foi estudado através de técnicas de Redes Neurais MLP e Redes Neurais de Base Radial (RBF) e testes foram feitos com modelos de duas e três classes. Por fim, a autora concluiu que o modelo de duas classes apresentou melhor resultado nas duas bases de dados analisadas e que este resultado é significativo, pois através desta abordagem, aumenta-se o número de registro por classes e consequentemente os algoritmos de classificação conseguem capturar melhor as estruturas inerentes nas amostras dos documentos pré-classificados. Sendo assim, todas as notícias utilizadas foram classificadas em positiva (quando a tendência do dia de negociação analisado é maior ou igual a zero) e negativa (quando a tendência do dia de negociação analisado é menor que zero).

Por fim, a partir da estratégia adotada acima (previsão um dia a frente) com modelos de duas classes, cada um dos documentos[6] (total de 337) da base de dados nprev, foi classificado em duas classes (positiva e negativa). A quantidade de documentos por classe ficou da seguinte forma: 156 (46,29%) documentos foram classificados como positivos e 181 (53,70%) documentos foram classificados como negativos.

6.2. Representação dos documentos

A proposta de representação dos documentos nesta tese é através de vetores *word embeddings* gerados a partir do algoritmo *word2vec*. Sendo assim, a base de dados composta pelo texto sem formatação (nw2v) obtida na seção de coleta de dados mencionada anteriormente, precisa ser processada de tal forma que a mesma possa ser utilizada no modelo do *word2vec*. Esta etapa pode ser definida em 2 passos fundamentais: o pré-processamento das notícias e a geração do modelo *word2vec*. Ambas detalhadas nas próximas seções (6.2.1 e 6.2.2).

[6] Um documento contém todos os títulos de notícias agrupados conforme as suas respectivas datas e o mesmo está relacionado a um dia de negociação da BmfBovespa.

6.2.1. Pré-processamento das notícias

O pré-processamento das notícias é uma tarefa importante que influencia diretamente os próximos passos das arquiteturas que envolvem as áreas de PLN. Na classificação de textos, que é um dos principais objetivos desta tese, um pré-processamento realizado erroneamente pode comprometer os resultados finais do modelo de previsão. Para (WANG *et. al*, 2005) o pré-processamento pode ser considerado como um processo de tornar claro cada estrutura da linguagem e eliminar, tanto quanto possível os fatores dependentes do idioma. Já (LEE *et al*., 2006) definiram esta tarefa apenas como remoção de *stop-words* e *stemming*. Em (CAMACHO-COLLADOS *et al*., 2017) os autores analisaram o impacto de simples decisões de pré-processamento no texto através do desempenho de modelos neurais convolucionais baseados em representação de palavras utilizando vetores *word embeddings*. Os autores afirmaram que de uma maneira geral e dependendo do domínio do conjunto de dados utilizados, um simples processo de tokenização funciona igual ou melhor que técnicas mais sofisticadas como a lematização (do inglês *lemmatization* ou *stemming*) ou o uso de agrupamento de múltiplas palavras-chave (N-grama). Além disto, os autores afirmaram também que o uso de vetores *word embeddings* tem melhorado muito o processo de generalização dos modelos de redes neurais (KIM, 2014; WEISS *et al*., 2015). Entretanto muitas das vezes os modelos recentes que fazem uso destes vetores não têm como foco um pré-processamento eficiente de seus conjuntos de dados, o que resulta em poucos estudos e análises sobre este tema. Devido ao fato da existência de poucos estudos envolvendo o pré-processamento na geração de modelos neurais e vetores *word embeddings*, nesta tese optou-se pela utilização de um pré-processamento envolvendo três tarefas fundamentais: (i) tokenização, (ii) limpeza dos dados e eliminação de stopwords e (iii) definição do modelo N-grama. Todas estas etapas foram executadas através de funções implementadas em *Python* e também utilizando a biblioteca NLTK.

Tokenização

Uma vez definido o conjunto de dados a ser utilizado no modelo, o próximo passo da metodologia proposta foi a etapa de pré-processamento (dividida em 3 passos fundamentais) conforme citado anteriormente. O primeiro passo nesta etapa foi o processo de tokenização que visa dividir ou quebrar o texto em palavras, frases, símbolos ou expressões compostas

por mais de uma palavra chamados tokens. A tokenização depende principalmente de heurísticas simples para separar os tokens seguindo alguns passos, ou seja, eles podem ser separados por tokens delimitadores como espaços em branco, sinais de pontuação ou quebras de linha. Sendo assim, todos os caracteres contidos em *strings* contínuas fazem parte de um token e os tokens delimitadores podem ou não ser utilizados no processamento. Por exemplo, analisando a sentença: Preços do petróleo devem subir gradualmente em 2017, aponta pesquisa. O processo de tokenização resultaria nos seguintes tokens: "Preços"; "do"; "petróleo"; "devem"; "subir"; "gradualmente"; "em"; "2017"; ","; "aponta"; "pesquisa"; ".". Se houver a necessidade do uso de tokens delimitadores como ponto final ("."), o mesmo pode ser utilizado, caso contrário ele pode ser excluído do processamento.

Limpeza dos dados e eliminação de *stopwords*

Uma vez definida a etapa de tokenização, a próxima tarefa correspondeu a limpeza de dados e a eliminação de *stopwords*. A limpeza de dados visa a remoção de números, caracteres não alfanuméricos, pontuações e espaços extras e também a alteração do texto para minúsculo. Em relação a eliminação de *stopwords*, o objetivo é remover palavras que não apresentam nenhum tipo de conhecimento nos textos. Geralmente as *stopwords* são adicionadas de forma manual a uma lista denominada *stoplist* (lista com palavras a serem descartadas no processamento). A maioria destas palavras são preposições, pronomes, artigos e outras classes de palavras auxiliares, que não contribuem para o processo de classificação e muitas das vezes por terem uma frequência muito grande em todos os idiomas, pode apresentar uma influência negativa nos resultados obtidos.

Dentro deste contexto, inicialmente foi necessário fazer duas cópias da base de dados nw2v antes da realização da remoção da pontuação dos textos (limpeza dos dados). Esta duplicação dos dados se deu devido ao fato de que para a execução da última etapa do pré-processamento (definição automática de n-grama) é necessário a utilização dos textos de notícias sem pontuação, entretanto para a tarefa de geração do modelo *word2vec*, o token delimitador ponto final é imprescindível para a definição do início e fim das sentenças. Logo, foram geradas duas bases de dados, uma sem pontuação e outra apenas com pontuações que fazem parte do final da frase como ("!", "?", e ".") que foram convertidos em ponto final. Por fim, ambas as bases de dados passaram pelo processo de remoção de *stopwords*, onde a lista de *stopwords* utilizada foi a *stoplist* que faz parte da biblioteca NLTK para processamento de textos em português.

Definição do modelo N-Grama

A última etapa do pré-processamento está relacionada a definição de n-gramas, que pode ser considerado como uma sequência contínua de *n* itens (ou tokens) de uma determinada sequência do texto. Ou seja, dada uma sentença ou um texto (t) é possível construir uma lista de n-gramas a partir de (t), encontrando pares de palavras que ocorrem próximas umas das outras com uma determinada frequência. As palavras ou termos n-gramas encontrados podem ser uma única palavra (N=1, unigramas), duas palavras (N=2, bigramas), três palavras (N=3, trigramas) ou qualquer outro valor N. Em PLN é possível também utilizar o termo de *Collocations*[7] para definir n-grama, ou conforme citado por (SAG *et al*., 2002) uma associação estatisticamente significativa e arbitrária entre itens que co-ocorrem, o que inclui também a expressão *Multiword expressions*[8]. Em (LYSE *et al*., 2012) os autores afirmaram que a definição e o uso de termos n-gramas ou *collocations* é muito importante principalmente em aplicações envolvendo tradução automática, quando as mesmas não conseguem ser determinadas composicionalmente a partir de palavras individuais. Neste caso, o sistema precisa saber se uma sequência de palavras pode ser traduzida palavra por palavra ou se tem um significado especial, exigindo uma tradução particular, em virtude de ser uma *collocation*.

Nesta tese em específico optou-se pela definição de n-grama devido ao fato de que algumas palavras que co-ocorrem com uma certa frequência em textos do mercado financeiro (assim como na tradução automática) fazem mais sentido quando são analisadas juntas do que individualmente. Citando como exemplo a sentença do título: "Bovespa perde força e dólar opera em alta". As palavras chaves ou características obtidas após o processo de tokenização e eliminação de *stopwords* são: "Bovespa"; "perde"; "força"; "dólar"; "opera"; "alta". Individualmente estas palavras não expressam nenhuma informação relevante para o investidor. Mas se esta mesma sentença for analisada sobre o ponto de vista de n-grama com N igual a 3, o resultado seria apenas 2 tokens ("Bovespa_perde_força"; "dólar_ opera_ alta"). Este tipo de informação é muito mais valioso para o investidor e consequentemente alimentar

[7] Colocações (do inglês *collocations*) são expressões de múltiplas palavras que comumente co-ocorrem no texto.

[8] Expressões multi-palavras (*Multiword expressions*) são definidas como palavras que co-correm com tanta frequência que podem ser identificadas como uma única unidade linguística e seus significados geralmente não podem ser determinados composicionalmente a partir dos significados das palavras individuais.

o classificador com estas informações aumentam as chances da obtenção de resultados mais condizentes com a realidade dos mercados financeiros. Ademais, o uso de n-gramas ajuda a melhorar a qualidade dos tokens encontrados, além de reduzir também a dimensionalidade das palavras-chave existentes nos documentos.

A extração automática a partir dos textos analisados de possíveis candidatos a n-gramas pode ser realizada através de uma métrica de associação ou medida estatística que é responsável por calcular a força de associação entre os tokens em um n-grama (LYSE *et al*., 2012). Estas medidas estatísticas são calculadas usando várias contagens de co-ocorrência e frequência individual de um determinado n-grama gerando assim uma lista, onde cada n-grama é associado a um conjunto de frequências observadas e frequências estimadas resultando em uma tabela de contingência (TC). E as informações obtidas nesta tabela são utilizadas para gerar as medidas de associação. Na tabela 6.2 pode ser visualizada a representação de uma TC e sua notação para um modelo de n-grama com n=2 (bigrama).

Tabela 6.2 - Tabela de Contingência de bigrama (token1 e token2) para frequências observadas

	token2	**#token2**	**Total**
token1	O_{11}	O_{12}	O_{1p} = R1
#token1	O_{21}	O_{22}	O_{2p} = R2
Total	O_{p1} = C1	O_{2p} = C2	O_{pp} = N

Analisando a tabela 6.2, cada frequência observada em um bigrama na TC é representada como um valor numérico O_{ij}, onde *i* e *j* representam a presença (valor=1) ou a ausência (valor=0) de cada token que representa o token bigrama. Sendo assim, o valor O_{11} indica quantas vezes o termo bigrama (token1 e token2) ocorreu no *corpus* e O_{12} indica quantas vezes token1 ocorreu na primeira posição, mas o token2 não ocorreu na segunda posição. As frequências da linha e coluna denominada "Total" representam as somas de cada linha. Sendo assim, O_{1p} (ou R1) é a soma de O_{11} e O_{12} e O_{p1} (ou C1) é a soma de O_{11} e O_{21} e assim sucessivamente. Já O_{pp} (ou N) é a soma total de todas as frequências nas margens da tabela (R1, R2, C1 e C2) que representa o número total de todos os n-gramas encontrados no *corpus*.

A TC é gerada a partir de uma tabela de frequência observadas e uma outra de frequências estimadas, sendo que esta última fornece as frequências esperadas, dada a

hipótese nula de que não há associação entre as palavras no n-grama analisado. Sendo assim, se não houver associação entre o token1 e token2 a chance de vê-los juntos é proporcional à frequência de cada um dos tokens individualmente. Logo, se um ou ambos os tokens forem altamente frequentes, então pode-se esperar uma alta frequência para e_{11} estimado também. A TC de frequências estimadas é apresentada na tabela 6.3.

Tabela 6.3 - Tabela de Contingência de bigrama (token1 e token2) para frequências estimadas

	token2	#token2
token1	$e_{11} = \frac{R1C1}{N}$	$e_{12} = \frac{R1C2}{N}$
#token1	$e_{21} = \frac{R2C1}{N}$	$e_{22} = \frac{R2C2}{N}$

As TCs podem ser geradas a partir de n-grama de qualquer tamanho. Entretanto, conforme se aumenta o valor de N, as TCs ficam mais complicadas porque o número de contagens das margens da tabela aumenta na ordem de 2^n. Nesta tese foram utilizados valores de n igual a 2 e 3, ou seja, termos bigramas e trigramas. Sendo assim, a fim de facilitar o entendimento da TC de trigramas, um exemplo da mesma pode ser visualizado na tabela 6.4. Já nas tabelas 6.5 e 6.6 estão representadas respectivamente as frequências das margens da tabela e frequências estimadas. Todas as tabelas relacionadas a TC tanto de bigramas como trigramas foram adaptadas de (LYSE *et al*., 2012).

Tabela 6.4 - Tabela de Contingência de trigrama (token1, token2 e token3) para frequências observadas

		token3	#token3
token1	token2	O_{111}	O_{112}
token1	#token2	O_{121}	O_{122}
#token1	token2	O_{211}	O_{212}
#token1	#token2	O_{221}	O_{222}

Assim como na TC de bigrama, na tabela 6.4 a célula O_{111} contém a frequência do token1, token2 e token3 ocorrendo juntos em suas respectivas posições. A célula O_{121} contém

a frequência da qual o token1 e token3 ocorrem em suas respectivas posições, mas o token2 não e assim sucessivamente para as demais células da tabela.

Tabela 6.5 - Frequência das margens da TC de trigrama (token1, token2 e token3)

O_{1pp}, O_{p1p}, O_{pp1}	número de trigramas onde o primeiro token é token1,token2 e token3 respectivamente
O_{2pp}, O_{p2p}, O_{pp2}	número de trigramas onde o primeiro token não é token1,token2 e token3 respectivamente
O_{11p}, O_{1p1}, O_{p11}	número de trigramas onde o primeiro e o segundo token; primeiro e terceiro token e segundo e terceiro token são respectivamente token1-token2; token1-token3 e token2-token3
O_{ppp} ou N	número total de ocorrências de todos os trigramas no corpus

Tabela 6.6 - Tabela de Contingência de trigrama (token1, token2 e token3) para frequências estimadas

$$e_{111} = \frac{O_{1pp} * O_{p1p} * O_{pp1}}{(N)^2} \qquad e_{222} = \frac{O_{2pp} * O_{p2p} * O_{pp2}}{(N)^2} \qquad e_{211} = \frac{O_{2pp} * O_{p1p} * O_{pp1}}{(N)^2}$$

$$e_{112} = \frac{O_{1pp} * O_{p1p} * O_{pp2}}{(N)^2} \qquad e_{121} = \frac{O_{1pp} * O_{p2p} * O_{pp1}}{(N)^2} \qquad e_{221} = \frac{O_{2pp} * O_{p2p} * O_{pp1}}{(N)^2}$$

$$e_{122} = \frac{O_{1pp} * O_{p2p} * O_{pp2}}{(N)^2} \qquad e_{212} = \frac{O_{2pp} * O_{p1p} * O_{pp2}}{(N)^2}$$

Uma vez definidas as informações das TCs de frequências observadas e estimadas é possível então calcular as medidas de associação que vão pontuar ou ranquear um determinado n-grama. Um exemplo de medida de associação comumente utilizada em abordagens envolvendo a extração de n-gramas é a medida de Informação Mútua Pontual (PMI) que classifica n-gramas comparando a frequência do candidato a n-grama à frequência dos componentes da expressão. Na equação 6.2 apresentada a seguir é possível visualizar o cálculo da medida de PMI para um bigrama, onde o valor O_{11} está relacionado a frequência com que a sequência ocorre, enquanto o valor estimado da sequência é baseado na frequência no qual cada uma das duas palavras que compõe o bigrama ocorre independentemente.

$$PMI = \log\left(\frac{O_{11}}{e_{11}}\right) \tag{6.2}$$

Para o cálculo da medida PMI para um trigrama a expressão a ser utilizada está apresentada na equação 6.3.

$$PMI = \log_2\left(\frac{O_{111}}{e_{111}}\right) \tag{6.3}$$

Várias outras medidas de associação foram sugeridas na literatura para a tarefa de extrair candidatos a n-gramas, como por exemplo os trabalhos de (CHURCH *et al*., 1989; DUNNING, 1993 e PEDERSEN, 1996). Segundo (EVERT, 2005) não existe uma lista razoavelmente abrangente do grande número de medidas de associações disponíveis. PEARCE (2002) fez uma avaliação de várias técnicas visando a extração automática de bigramas. No experimento, para cada técnica testada foi fornecida as frequências de co-ocorrência de bigramas obtidos a partir de milhões de palavras. Para cada par de palavras nestes dados de frequência, a técnica foi usada para atribuir uma pontuação, no qual quanto maior a pontuação melhor o bigrama. Alguns critérios pré-definidos (como aplicação de um limiar e ignorar algum tipo de pontuação) foram utilizados para melhorar a precisão da técnica adotada. Equações bem definidas com cálculos e exemplos foram apresentadas pelo autor neste trabalho. SCHONE *et al*. (2001) também apresentaram um trabalho com equações bem detalhadas e concisas para avaliar nove medidas de associação diferentes. BREIDT (1993) avaliou uma combinação das medidas de Informação mútua (MI) e *T-Score* para a extração de bigramas na língua alemã. Em (LYSE *et al*., 2012) foram aplicadas várias medidas de associação na sequência de palavras do *Norwegian Newspaper Corpus*[9] (NNC) visando a classificação de palavras bigramas e trigramas em termos de sua tendência para co-ocorrer. Nove medidas de associação foram utilizadas na definição dos bigramas. Entretanto, em relação a definição de trigramas, os autores reportaram que os cálculos não são diretos quando se trata de sequências maiores que bigramas. Sendo assim, os autores fizeram uso de apenas 4 medidas de associação (Razão de verossimilhança de log, Informação Mútua, Informação Mútua Pontual e *Poisson Stirling*) seguindo a sugestão de (BANERJEE *et al*., 2003).

Dentro deste contexto é possível destacar que a escolha pela métrica ideal a ser usada em uma determinada aplicação ainda é um motivo de muita pesquisa na literatura. Nesta tese, conforme citado anteriormente a definição de n-gramas foi implementada no *Python* utilizando o pacote *Collocations*[10] da biblioteca NLTK. Esta biblioteca disponibiliza várias medidas de associação para esta finalidade, a saber: Informação Mútua Pontual (*Pointwise*

[9] http://uni.no/en/uni-computing/clu/the-norwegian-newspaper-corpus/
[10] http://www.nltk.org/howto/collocations.html

Mutual Information - PMI), Chi-quadrada, Razão de Verossimilhança (*Likelihood Ratio*), *Jaccard*, *Mi-like*, *Poisson-Stirling* e *T-Student*. Seguindo os trabalhos citados anteriormente, optou-se por utilizar todas as medidas de associação disponíveis no NLTK e fazer uma comparação dos resultados das mesmas, ou seja, validar o uso de todas as medidas utilizadas apenas na etapa final da previsão. Logo, nenhuma regra foi definida para decidir qual das métricas seria utilizada. Detalhes de todas as métricas citadas assim como a implementação das mesmas em *Python* podem ser encontradas em (BIRD *et al*., 2009).

Uma vez estabelecidas as métricas de associação, foi preciso definir a quantidade de palavras que co-ocorrem (valor de *n*) no texto. Inicialmente foram feitos alguns testes com n=2 (bigramas) e n=3 (trigramas). Sendo assim, ao processar os textos da base nw2v sem pontuação (tokens gerados após a etapa de tokenização e limpeza dos dados) com as métricas sugeridas e com n=2 e n=3 foram gerados milhares de bigramas e trigramas. A fim de melhorar a qualidade destas palavras foi utilizado um processo de filtragem, onde foi definido um limiar mínimo (ω) (foram testados dois limiares: 50 e 100) para a frequência de co-ocorrência. Portanto, somente bigramas e trigramas que apareceram juntos mais de 50 ou 100 vezes foram considerados no processamento. Com este processo de filtragem a quantidade de bigramas gerados ficaram em torno de 1750 e 4320 para os filtros 100 e 50 respectivamente. Já em relação aos trigramas, os mesmos reduziram de milhares para a ordem de centenas, algo em torno de 350 e 950 possíveis trigramas com limiares de 100 e 50 respectivamente.

Fazendo uma análise manual das listas de possíveis bigramas encontrados através das métricas utilizadas verificou-se que muitos deles não faziam muito sentido dentro do contexto analisado, logo optou-se por utilizar apenas palavras trigramas, ou seja, n igual a 3.

Sendo assim, após a definição dos trigramas para todas as métricas utilizadas, os mesmos foram substituídos nos textos de notícias resultantes do processamento das etapas anteriores, neste caso foram utilizados os textos da base de dados nw2v com pontuação. Foram gerados então 14 arquivos de textos de notícias, onde cada arquivo corresponde aos textos com seus respectivos trigramas encontrados por uma determinada medida de associação (7 medidas) e um determinado processo de filtragem (ω = 50 e 100). Na tabela 6.7 é possível visualizar alguns trigramas obtidos com filtro ω=100 e métrica de associação Informação Mútua Pontual (PMI).

Tabela 6.7 - Trigramas obtidos após o processamento dos textos com a métrica PMI

Trigramas: Informação Mútua Pontual (Métrica)		
('dólar', 'fecha', 'alta')	('dólar', 'abre', 'queda')	('bovespa', 'opera', 'queda')
('bolsas', 'europa', 'caem')	('petróleo', 'opera', 'alta')	('ferro', 'fecha', 'alta')
('dólar', 'opera', 'baixa')	('dólar', 'leve', 'queda')	('europa', 'fecham', 'queda')
('dólar', 'cai', 'mais')	('alta', 'juros', 'eua')	('desemprego', 'eua', 'sobem')
('ibovespa', 'opera', 'alta')	('petróleo', 'opera', 'queda')	('pib', 'eua', 'cresce')

6.2.2. Geração do Modelo Word2Vec

A proposta de representação dos documentos nesta tese é através de vetores *word embeddings* gerados a partir do algoritmo *word2vec*. Sendo assim, as bases de dados geradas a partir da execução das medidas de associação (14 arquivos gerados na etapa anterior) foram processadas utilizando o algoritmo *word2vec* na versão *Skip-gram*. A ferramenta utilizada para o objetivo desta etapa foi a *Gensim* (RADIM, 2010), que é uma biblioteca livre (gratuita tanto para uso pessoal como para uso comercial) implementada em *Python* e responsável por analisar a estrutura semântica de documentos textuais simples e recuperação de documentos semanticamente similares. É uma ferramenta multiplataforma, que pode processar uma grande quantidade de dados e também muito utilizada pelas comunidades de PLN e Recuperação de Informação (IR), pois apresenta diversos tutoriais disponíveis na internet, o que facilita o uso e o entendimento da mesma. Seus principais algoritmos fazem uso de rotinas matemáticas altamente otimizadas que podem ser usadas em paralelo e a mesma já tem a implementação da versão do algoritmo proposto neste trabalho (*word2vec* na versão *Skip-gram*).

Conforme citado, para cada métrica de associação utilizada e seu respectivo limiar de frequência (50 e 100) foi gerado um arquivo “.txt”’, totalizando 14 arquivos contendo todos os títulos de notícias no período de 04/01/2012 a 29/12/2016 (em média 4823507 tokens), onde tokens trigramas foram substituídas e os caracteres “? e “!” também foram substituídos por ponto final (.) e todos os textos foram colocados em formato de sentenças a fim de serem entendidos pela ferramenta e utilizados no treinamento do algoritmo *word2vec*.

Em seguida, o algoritmo *word2vec* na versão *skip-gram* foi utilizado para gerar os vetores *word embeddings* para cada sentença de cada texto ".txt" (*input* do algoritmo) gerado anteriormente. A versão do *word2vec skip-gram* disponível na ferramenta *Gensim* consta de vários parâmetros a serem definidos, dentre os quais é possível destacar: (i) ***size*** = dimensionalidade do vetor denso ou vetor característica (quantidade de neurônios da camada escondida do modelo RNA) utilizado para representar cada token ou palavra; (ii) ***window =*** tamanho do contexto (distância máxima entre a palavra alvo e seus vizinhos); (iii) ***min_count*** = frequência mínima de contagens de palavras, ou seja, o modelo ignora palavras que não satisfazem a contagem mínima definida; (iv) ***workers*** = quantidade de cores a ser utilizado no treinamento e (v) ***sg*** = define o algoritmo de treinamento (sg=0 para modelo *Cbow* ou sg=1 para modelo *skip-gram*). A configuração final de parâmetros proposta foi a seguinte: *size* = 100 e 300; *window* = 5; *min_count* = 50; *workers* = 32 e *sg* = 1.

Após a execução do *word2vec* foram gerados 28 vocabulários (ou tabela de vetores *word embeddings*) contendo tokens unigramas e trigramas, onde cada um destes vocabulários apresentou as seguintes características: uma determinada métrica de associação, um limiar de filtragem ω e uma determinada dimensão *d*, ou seja, dentro do contexto de parâmetros do modelo, foram 7 métricas de associação com dois limiares de filtragem e 2 tamanhos de dimensão do vetor *word embedding* (parâmetro *size* do modelo) totalizando 28 tabelas de vetores *word embeddings*.

Vale ressaltar, que o treinamento do *word2vec* é uma tarefa não supervisionada, o que dificulta uma avaliação objetiva dos resultados. Muitas das vezes esta avaliação depende muito do resultado final da tarefa de PLN executada. SCHNABEL *et al.* (2015) afirmaram que apesar do crescente interesse em representações vetoriais *word embeddings*, ainda são poucos os trabalhos que fazem avaliações diretas destes modelos. Neste mesmo trabalho, os autores destacaram algumas abordagens para medir a qualidade de vetores *word embeddings* fazendo uma análise abrangente da avaliação de métodos que podem ser utilizados para esta finalidade. Quatro métodos foram analisados pelos autores, dentro dos quais é possível citar a métrica de analogia que foi proposta inicialmente por (MIKOLOV *et al*., 2013b) e a de relacionamento, sendo esta última utilizada nesta tese para avaliar o modelo proposto. A métrica de relacionamento visa encontrar uma correlação entre palavras através da similaridade do cosseno comparável a pontuações sugeridas por especialistas humanos. Sendo assim, algumas palavras foram escolhidas nos textos utilizados nesta tese e foi realizada uma busca pelas 30 palavras mais similares às mesmas, utilizando a medida da similaridade do cosseno. De uma maneira geral, os resultados encontrados realmente

apresentaram correlações fortes entre palavras similares. Na figura 6.5 pode ser visualizado o resultado obtido para uma destas palavras analisadas (bovespa) situada no centro do grafo apresentado, onde é possível identificar que todas as palavras relacionadas tendem a ser utilizadas em contextos semelhantes. Na tabela 6.8 também pode ser observado o mesmo resultado para a palavra bovespa, porém com a apresentação dos scores para algumas palavras similares encontradas.

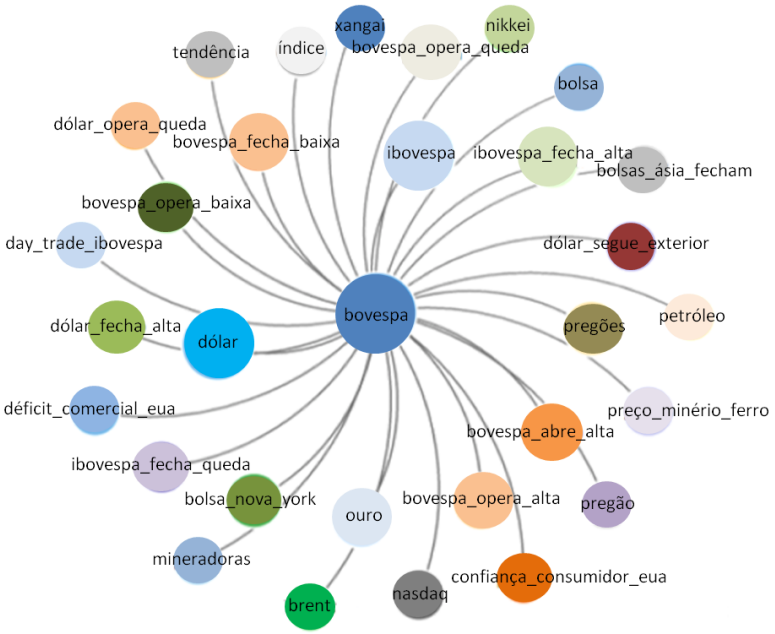

Figura 6.5 - As 30 palavras mais similares à palavra chave Bovespa após processamento do algoritmo word2vec

Tabela 6.8 - Palavras mais similares à palavra chave Bovespa após processamento do algoritmo word2vec e seus *scores*

Palavra similar	Pontuação
ibovespa	0.8470997214317322
dólar	0.648729681968689
ouro	0.6447665691375732
bovespa_opera_alta	0.6390209794044495
pregões	0.6189913749694824
Ibovespa_fecha_alta	0.6176074743270874
bolsa_nova_york	0.6119552850723267
⋮	⋮
mineradoras	0.45871585607528687

Após a finalização do treinamento e da geração dos vetores *word embeddings* a partir do algoritmo *word2vec*, o passo seguinte compreendeu em transformar a base de dados nprev (utilizada para a etapa de previsão do Ibovespa) em vetores sentenças. Sendo assim, as sentenças (ou títulos) da base de dados nprev foram organizadas sequencialmente e em ordem cronológica por dia de publicação do título da notícia, ou seja, cada sequência corresponde a um conjunto de títulos de notícias de um determinado dia (t). O próximo passo visou a codificação destas sentenças que consistiu em calcular a média aritmética dos vetores *word embeddings* de todos os tokens (palavras-chaves) constituintes de uma determinada sentença, gerando assim um vetor que foi denominado como vetor sentença (*sentence embedding*), conforme apresentado na figura 6.6. Trabalhos recentes (KENTER *et al.*, 2016) têm demonstrado que o uso destes vetores (*sentence embedding*) pode ser uma característica forte em várias tarefas de PLN.

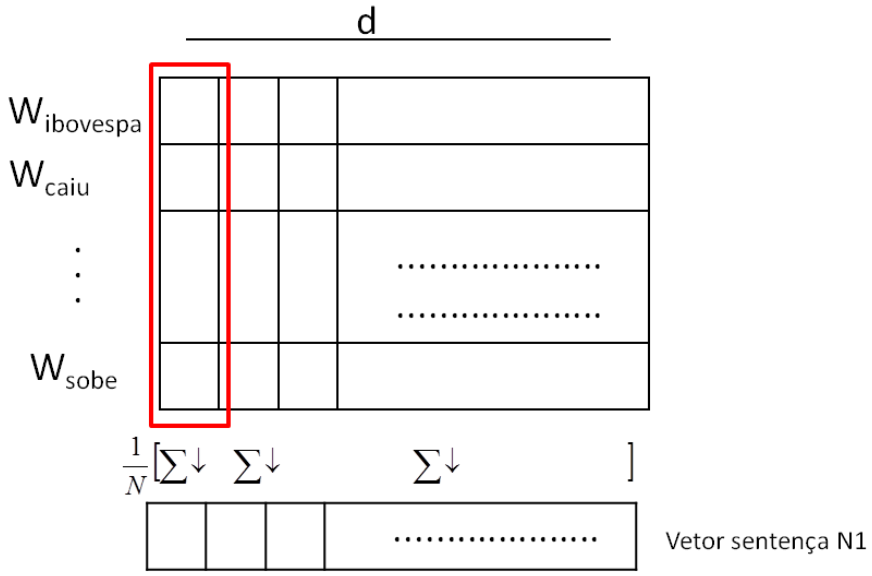

Figura 6.6 - Representação Vetor Sentença (*sentence embedding*)

Na figura apresentada anteriormente foi usado como exemplo a sentença (ou título) já pré-processada “Ibovespa caiu rumores demissão presidente Petrobrás dólar sobe”. Cada token da sentença está representado na tabela por uma das possíveis representações vetoriais *word embeddings* da mesma (vetor de pesos W), gerada após o treinamento do algoritmo *word2vec*, *d* representa a dimensão deste vetor (foram testadas dimensões com valores de 100 e 300) e N está associada a quantidade de tokens que fazem parte de uma sentença. A partir dos vetores gerados (*sentence embedding*) foi possível criar uma tabela (*lookup table*) para cada dia de negociação do Ibovespa que serviu como entrada para o modelo de rede neural CNN responsável por aprender padrões a partir de representações de características complexas.

6.3. Extração do conhecimento e classificação das notícias

Após a execução da tarefa final do pré-processamento que transformou a versão textual dos documentos (ou títulos de notícias) em vetores de sentenças (*sentence embeddings)*, o passo seguinte está relacionado ao processo de classificação destes vetores. Para esta finalidade foi construído um modelo de rede neural CNN com diferentes camadas de processamentos juntamente com as técnicas de redes neurais de Aprendizado Extremo para tentar quantificar, predizer ou mesmo aprender o relacionamento entre o conteúdo dos títulos das notícias e a tendência diária do comportamento do índice (Ibovespa) do mercado acionário brasileiro (BmfBobespa - Bolsa de Valores, Mercadorias e Futuros).

Na abordagem de classificação de textos os documentos já previamente classificados são divididos em dois ou três subconjuntos distintos, definidos como treinamento e teste, ou treinamento, teste e validação. O conjunto de treinamento é utilizado para treinar o classificador e encontrar padrões nos documentos, enquanto o conjunto de teste (não rotulado) é utilizado para avaliar o desempenho do modelo. Já o conjunto de validação (também rotulado) visa verificar a eficiência da rede quanto à sua capacidade de generalização durante o treinamento da mesma. Neste caso, o conjunto de validação pode ser empregado como critério de parada do treinamento. Os resultados obtidos após o treinamento podem ser medidos através de uma taxa de acerto do modelo, comparando os resultados obtidos com a rotulação disponível na base de testes.

Estes subconjuntos geralmente precisam ser distintos para garantir que as medidas obtidas utilizando o conjunto de teste sejam de um conjunto diferente daquele utilizado para realizar o aprendizado (conjunto de treinamento), tornando assim a medida estatisticamente válida. Existem vários métodos que podem ser utilizados para realizar esta subdivisão dos dados garantindo assim que os resultados obtidos sejam estatisticamente válidos e que o modelo proposto apresente um bom desempenho na sua capacidade de generalização. Dentre estes métodos é possível citar o *Holdout*, *Leave One-Out* e *K-Fold Cross Validation*, sendo este último considerado como o mais popular (ARLOT & CELISSE, 2010) e escolhido para ser utilizado nos experimentos realizados nesta tese. O método de *K-Fold Cross Validation* (ou validação cruzada de *k* partições) consiste de três passos fundamentais: a base de dados inicialmente é dividida em *k* subconjuntos aproximadamente iguais; em seguida *K-1*

subconjuntos são utilizados para treinamento e o subconjunto restante é utilizado para testar; estes dois procedimentos são repetidos *K* vezes (neste trabalho foi definido *k* igual a 10) utilizando sempre um subconjunto diferente para testar. A consequência disto é que todos os documentos estão disponíveis para teste. O resultado final é obtido pela média dos resultados em cada etapa. A fim de melhorar a capacidade de generalização do modelo, 10% dos dados que fazem parte do conjunto de treinamento foram retirados do mesmo e utilizados para validar o treinamento. Logo, o modelo apresentou dois critérios de parada, o número máximo de épocas[11] e a taxa de generalização da rede obtida através do conjunto de validação em algum momento do treinamento da rede, ou seja, o treinamento foi interrompido quando um ou outro critério foi atendido.

Uma vez definidos os subconjuntos citados, os mesmos foram utilizados como entrada para o modelo de classificação proposto (CNN + ELM).

As redes CNNs conforme citado na seção 3.3 deste trabalho, geralmente são compostas de uma camada de entrada, camadas convolucionais, camadas de *pooling* e camadas totalmente conectada. Muitos estudos têm sido propostos na literatura com diferentes configurações de parâmetros e os mesmos têm demonstrado que as CNNs podem ser utilizadas com sucesso em muitas aplicações envolvendo aplicações de PLN e classificação de textos. De fato, (KALCHBRENNER *et al*., 2014) obteve um excelente desempenho na classificação de textos utilizando uma CNN com várias camadas convolucionais. Já (KIM, 2014) reportou uma série de experimentos com bons resultados visando análise de sentimentos e classificação de textos utilizando uma arquitetura CNN relativamente simples, com uma camada de convolução com múltiplos filtros, uma camada de *pooling* usando o operador *max* e o classificador *softmax*. Em (ZHANG *et al*., 2015) os autores fizeram uma avaliação empírica do efeito da utilização de diferentes parâmetros nas arquiteturas da rede CNN para classificação de textos, investigando o impacto e performance na variação destes parâmetros em múltiplas execuções. Alguns pontos importantes apontados pelos autores em suas conclusões foram: a eficiência do operador de *max* (*max pooling*) quando comparado com a média (*average pooling*); a importância das dimensões dos filtros e o fato dos mesmos serem totalmente dependentes da tarefa a ser executada.

Dentro deste contexto, a arquitetura CNN proposta nesta tese é composta de apenas uma camada convolucional com múltiplos filtros e uma camada de *pooling* com operador *max*. Este tipo de arquitetura conforme citado em (KIM, 2014) tem demostrado ser tão

[11] Épocas: número máximo de ciclos realizados durante o treinamento de uma rede neural ou outro método de classificação.

eficiente quanto o uso de múltiplas camadas convolucionais. Ademais, a incorporação de mais camadas de processamento pode demandar mais parâmetros a serem configurados e consequentemente mais tempo de treinamento. Na figura 6.7 apresentada a seguir pode ser visualizado a arquitetura proposta.

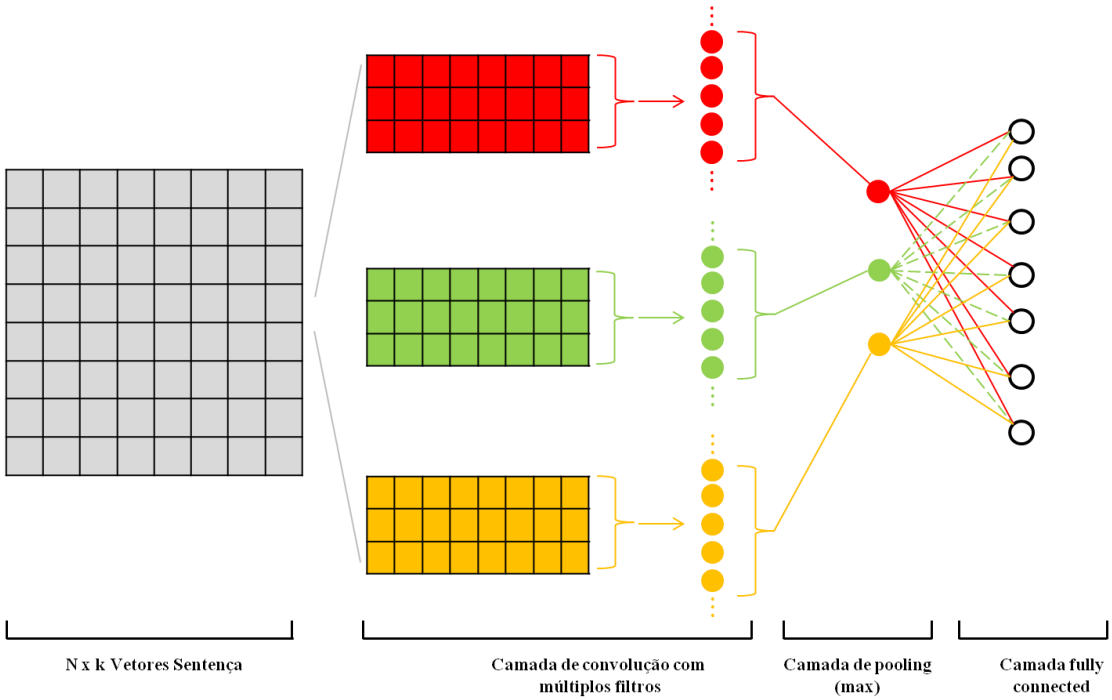

Figura 6.7 - Arquitetura CNN proposta

A camada de entrada (ou *input*) da rede CNN é uma tabela contendo uma sequência de títulos de notícias representadas pela média de seus vetores *word embeddings* ou vetores *sentence embeddings* [w_1, w_2,...,w_N], onde $w_i \in R^k$, N está associado a quantidade de títulos de notícias em um dia de negociação do Ibovespa e *k* é a dimensão do vetor *sentence embedding*. A CNN requer entradas com tamanhos fixos e a quantidade de notícias diárias pode variar muito. Logo a definição do valor máximo de notícias diárias a ser utilizado no modelo (valor de 450 notícias) foi determinado baseado no valor máximo da distribuição de notícias (histograma) diárias encontradas na base de dados analisada (nprev) após a etapa de pré-processamento, conforme pode ser observado na figura 6.8.

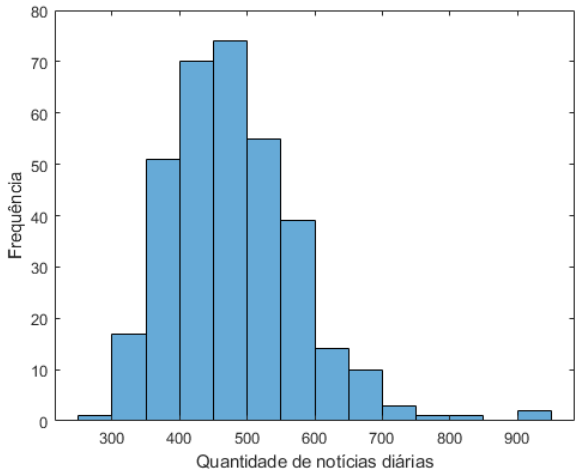

Figura 6.8 - Distribuição da quantidade de notícias diárias contidas em toda a base de dados analisada depois do pré-processamento

Como foram testadas duas dimensões diferentes de vetores *sentence embeddings* (k=100 e k=300), o tamanho da matriz de entrada da CNN apresentou as seguintes resoluções: 450 x 100 e 450 x 300, onde o valor de 450 está relacionado a quantidade de notícias diárias em um determinado dia de negociação do Ibovespa e os valores 100 e 300 representam a dimensão do vetor *sentence embedding*. Se em um determinado dia o número de notícias foi menor que 450, zeros foram adicionadas ao final da tabela para atender ao tamanho da matriz de entrada. Para dias em que o número de notícias foi maior que 450, somente as 450 primeiras notícias foram consideradas representativas para este dia analisado.

O objetivo da camada convolucional da CNN é extrair padrões e características semânticas presentes nos textos através da combinação de vetores *sentence embeddings* em uma janela. A operação de convolução envolve um filtro $F \in R^{hk}$, no qual é aplicado uma janela de tamanho *h* para produzir um novo conjunto de características. Estes filtros F são considerados como vetores pesos da camada convolucional. O cálculo para gerar uma nova característica y_i a partir de uma janela de sentenças $x_{i:i+h-1}$ é dada pela equação:

$$y_i = f(F.x_{i:i+h-1} + b) \tag{6.4}$$

Onde $b \in R$ e está relacionado ao termo bias e *f* é a função de ativação não linear. Esta operação é aplicada a cada possível janela na sentença para produzir o mapa de característica $y = [y_1, y_2, ..., y_{n-h+1}]$ com $y \in R^{n-h+1}$. A função de ativação utilizada aqui foi a função ReLu apresentada na equação 2.8.

Sendo assim, através de um único filtro é possível transformar uma matriz de vetores *sentence embeddings* em um mapa de características e o uso de vários filtros em paralelo

pode dar a esta matriz uma representação mais rica em informações (FENG *et al*., 2018). Esta transformação pode ser visualizada na arquitetura da CNN apesentada na figura 6.7, onde através de 3 filtros com tamanho de janela (h x k), onde h é igual a 3 e k está associado a dimensão vetor *sentence embedding*, são produzidos 3 mapas de características.

Ademais, ainda dentro do conceito da camada convolucional são necessários a definição de mais dois hiperparâmetros: *wide* ou *narrow convolution* e o *stride*, ambos detalhados na seção 3.3.1 deste trabalho. Sendo assim a configuração dos parâmetros da camada convolucional sugerida nesta tese foi definida da seguinte forma: a quantidade de filtros variou de 32 a 160, mais precisamente foram testados valores de 32, 64, 96, 128 e 160 filtros com tamanho fixo de 3 x k, 5 x k e 7 x k, onde k representa a dimensão do vetor *sentence embedding* ; *stride* igual a 1 e não utilização do *padding* (*narrow convolution*) para todas as possíveis configurações de filtros.

As saídas da camada de convolução foram utilizadas como entrada para a camada de *pooling* (detalhada na seção 3.3.2) que é responsável por obter as características de maior relevância através de métodos estatísticos, tais como o operador de máximo, média e norma L2. Para os experimentos realizados nesta tese foi utilizado o operador de máximo (*max*) que permite a seleção de características semânticas globais e tenta capturar as características mais importantes através dos valores de máximos dos mapas de características. Isto é, para cada mapa de características gerados na camada convolucional foi utilizado um único valor (*max*) pra representar aquele filtro. E os valores de máximos (*max*) de todos os filtros convolucionais utilizados foram concatenados formando um vetor de característica unidimensional.

A partir dos resultados obtidos no processamento das camadas anteriores foi gerado um vetor característica na última camada da rede CNN (*fully connected*) e o mesmo foi utilizado como entrada para a rede neural ELM responsável pelo treinamento e classificação dos dados. Detalhes sobre a ELM, assim como a motivação para a utilização da mesma podem ser encontradas na seção 3.4. O algoritmo da ELM utilizado foi uma versão proposta por (LIANG *et al*., 2006) denominada *Online Sequencial Extreme Learning Machine* (OS-ELM). A função de ativação utilizada foi a função sigmoide apresentada na equação 3.6 e a quantidade de neurônios da camada escondida (L), único parâmetro variado, foi definida conforme proposto em (CAO *et al*., 2017), ou seja, através da seguinte equação:

$$L = K * N \tag{6.5}$$

Onde N está relacionado à quantidade de registros do conjunto de treinamento e K assumiu os valores de 0.2, 0.4 e 0.6.

Após o treinamento da rede (etapa final do processo de classificação), o conjunto de teste foi utilizado para validar o modelo. Os resultados finais foram armazenados em uma matriz, denominada matriz confusão[12], onde foi possível contabilizar as taxas de acertos e erros do modelo para cada classe analisada. Detalhes do cálculo da matriz confusão, assim como das medidas de erro associadas à mesma podem ser encontrados em (GOADRICH *et al*., 2006).

A implementação desta última etapa da metodologia proposta (processo de classificação) também foi realizada utilizando a linguagem de programação *Python*, entretanto algumas bibliotecas foram adicionadas para facilitar no processo de implementação, a saber: biblioteca de *deep learning keras*[13], juntamente com a biblioteca *TensorFlow*[14], *Scikit-Learn*[15], e a versão online do algoritmo OS-ELM[16], porém com algumas modificações para atender aos requisitos da metodologia proposta.

6.4. Comparação e variações de modelos

Conforme citado na introdução deste trabalho a principal proposta do mesmo foi o aprofundamento e investigação das potencialidades de duas arquiteturas de redes neurais (CNN + ELM) juntamente com técnicas de PLN para predizer a tendência diária do comportamento futuro do Ibovespa. Toda a metodologia utilizada para alcançar este objetivo foi descrita nas seções apresentadas anteriormente neste mesmo capítulo. Entretanto a fim de avaliar melhor a influência das arquiteturas propostas aqui e do método de representação de documentos através dos vetores *sentence embeddings* foi realizada uma comparação entre o modelo proposto neste trabalho, o I-CNNELM e dois outros modelos onde as arquiteturas de rede CNN (I-CNN) e ELM (I-ELM) foram utilizadas separadamente visando o mesmo objetivo desta tese. Ademais, os resultados do modelo proposto (I-CNNELM) também foram

[12] Matriz confusão: Utilizada para avaliar a acurácia de um classificador. Cada linha da matriz representa as instâncias de uma classe real, enquanto cada coluna representa as instâncias em uma classe prevista. Várias medidas podem ser derivadas a partir dos valores da matriz confusão como acurácia, precisão, *recall* e a medida F (*F-Measure*).
[13] https://keras.io/
[14] https://www.tensorflow.org/
[15] http://scikit-learn.org/stable/#
[16] https://github.com/otenim/TensorFlow-OS-ELM

comparados com os resultados obtidos através de outro modelo (*Bag-of-Keywords* – BoK) reportado na literatura e proposto por (PENG *et al*., 2015) onde o algoritmo de classificação utilizado foi uma rede neural MLP com múltiplas camadas e a representação das palavras foi realizada utilizando o modelo *bag-of-words* citado na seção 4.1

Sendo assim, o primeiro modelo (I-CNN) fez uso do mesmo conjunto de dados e das mesmas técnicas de pré-processamento das notícias. Entretanto a etapa de classificação foi realizada utilizando a rede neural CNN juntamente com o algoritmo de treinamento gradiente descente estocástico (*Stochastic Gradient Descent* – SGD), taxa *momentum* que permite controlar a velocidade das mudanças dos parâmetros da rede com valor de 0.8 e taxa de aprendizado com decaimento exponencial no valor de 0.1. Para as camadas convolucionais e *pooling* foram mantidas as mesmas configurações propostas no modelo I-CNNELM. Foi adicionada também neste modelo aos resultados da camada de *pooling*, uma função *dropout* com um valor de probabilidade igual a 0.5, para ajudar a evitar o *overfitting* e melhorar o poder de generalização do modelo conforme explicado na seção 3.3.3.

Nos dois outros modelos (I-ELM e I-BoK), na etapa de pré-processamento, diferentemente dos modelos anteriores, foi utilizado a metodologia proposta por (PENG *et al*., 2015), detalhado nos trabalhos correlatos, onde a representação das palavras foi realizada utilizando o modelo *bag-of-words* com a medida TFIDF para a atribuição de pesos dos termos (ou palavras-chave). E estes termos foram determinados a partir dos vetores *word embeddings* gerados pelo algoritmo *word2vec* e por uma lista de palavras-chave relacionadas ao mercado financeiro e selecionadas manualmente. De uma forma mais ampla e dentro do contexto dos autores citados, este modelo de representação de palavras foi gerado da seguinte forma: primeiramente foram selecionados os vetores *word embeddings* de todas as palavras contidas no *corpus* (estes vetores são os mesmos gerados pelo algoritmo *word2vec* para o modelo proposto nesta tese). O passo seguinte foi a seleção de um pequeno conjunto de palavras-chave (unigramas e trigramas) que de alguma forma fazem parte do vocabulário diário de investidores que trabalham no mercado de ações. Na referência citada os autores fizeram uso de 9 palavras-chave, entretanto nesta tese optou-se por um número maior de palavras-chave, ou seja, 23 no total. Algumas destas palavras são: aumentou, caiu, cresceu, ações_europeias_caem, bolsas_Ásia_sobem, positivo, ganho, etc. No passo seguinte, com base nos vetores *word embeddings* selecionados, foram geradas novas palavras-chave baseadas no cálculo da distância do cosseno das 23 palavras-chave definidas. A distância do cosseno representa a semelhança entre duas palavras no espaço vetorial. Logo, para uma dada palavra-chave, como por exemplo, “positivo”, foram selecionadas várias outras palavras com

vetores *word embeddings* similares ao vetor *word embedding* que representa a palavra analisada "positivo". A ferramenta utilizada na geração destes vetores (*Gensim*) permite o cálculo da similaridade do cosseno para N vetores similares, sendo assim, o valor de N definido nesta tese foi de 30, ou seja, foram calculados os 30 vetores mais similares para cada vetor de palavra-chave definida manualmente. Desta forma, um vetor característica de 713 dimensões ou atributos (já incluídas as 23 palavras-chave definidas manualmente) foi gerado para cada amostra, onde cada dimensão deste vetor é representada pela medida TFIDF calculada a partir de cada palavra-chave selecionada em todo o *corpus* conforme pode ser visualizado no exemplo citado na tabela 6.9.

Tabela 6.9 - Exemplo de representação final dos documentos após a conversão tabela Atributo-Valor via TFIDF

	Atributo 1	Atributo2	...	Atributo 713	Classe
D1	0.2	0.21		0.24	1
D2	0.3	0.11		0.67	2
...	...	...		...	...
D337	0.76	0.12		0.54	1

Na tabela citada anteriormente o número de registros está relacionado à quantidade de documentos utilizados na base de dados nprev (total de 337 documentos); o número de atributos é igual à quantidade de palavras-chave obtidas após a implementação do pré-processamento citado nesta seção (vetores *word embeddings* + palavras-chave manuais) totalizando 713 palavras-chave; o peso destes atributos são os valores calculados pela medida TFIDF e a classe de cada registro está relacionada à classe (positiva ou negativa) atribuída a cada documento conforme estratégias citadas na seção 6.1.3.

Esta tabela foi então utilizada como entrada para os modelos de redes neurais ELM (modelo I-ELM) e MLP (modelo I-BoK). Para o modelo I-ELM foram alterados ao longo do treinamento a quantidade de neurônios da camada escondida da rede ELM. Os valores utilizados foram os mesmos propostos no modelo I-CNNELM, ou seja, 64, 135 e 200 neurônios. Já o modelo I-Bok foi estruturado com a mesma configuração sugerida pelos autores em (PENG *et al*., 2015), rede MLP com 4 camadas escondidas tendo 1024 neurônios cada uma.

Após a etapa de treinamento de todos os modelos de redes sugeridos nesta fase do trabalho (comparação de modelos), o conjunto de teste foi utilizado para validar os mesmos, tendo a matriz confusão como base para os cálculos das taxas de erros e acertos dos modelos.

Por fim, assim como no I-CNNELM, a implementação destes modelos também foi realizada utilizando o *Python* e a biblioteca de *deep learning keras*, juntamente com a biblioteca *TensorFlow*, *Scikit-Learn*, e a versão online do algoritmo OS-ELM.

6.5. Estratégia de Negociação

O Ibovespa tem contratos futuros negociados na BM&F (Ibovespa futuro) e são identificados pelo termo IND, seguido por letras indicando o mês de vencimento (G, J, M, Q, V, Z) e dois números referentes ao ano de vencimento (por exemplo, INDV16). Estes contratos futuros são chamados de contratos cheios ou padrão e são negociados diariamente no mercado BM&F. O valor de um contrato futuro do Ibovespa corresponde em reais (R$) ao número de pontos do contrato e o lote mínimo a ser negociado é de cinco contratos, ou seja, se o Ibovespa futuro para um vencimento específico está em 60 000 pontos, então seu valor será de 60 000 reais e o valor mínimo a ser negociado será de 300 000 reais. É possível verificar que o lote mínimo a ser negociado é um valor alto, o que dificulta a entrada do investidor que possui um menor capital nestes tipos de transações envolvendo a compra e venda de contratos futuros. Sendo assim, uma possível opção para estes investidores é a compra e venda de minicontratos do Ibovespa, que representam uma pequena parcela dos contratos cheios.

Da mesma forma dos contratos cheios, os minicontratos futuros são derivativos de compra e venda da estimativa do valor do indicador Ibovespa para uma data futura. Os mesmos são negociados no Mercado BM&F com referência nas mesmas datas de vencimento pré-definidas para o Ibovespa futuro. O código de negociação do minicontrato inclui o código WIN acrescido de uma letra (G, J, M, Q, V ou Z) indicando o mês de vencimento do contrato (conforme tabela 6.10), seguido de dois números correspondentes ao ano de vencimento. A data do vencimento é toda quarta-feira mais próxima do dia 15 dos meses pares. De uma forma geral, estes negócios visam aproveitar as oscilações das ações que compõem o Ibovespa.

Tabela 6.10 - Código de negociação minicontrato Ibovespa

Código	Mês de vencimento
G	Fevereiro
J	Abril
M	Junho
Q	Agosto
V	Outubro
Z	Dezembro

O lote padrão do minicontrato futuro corresponde a 1 minicontrato com um valor financeiro correspondente a um 1/5 (20%) do valor do contrato cheio. Por exemplo, se o Ibovespa futuro (para um mês específico) vale 70 000 pontos (esse contrato futuro cheio custa 70 000 reais) então o minicontrato custará 14 000 reais, ou seja, a cotação de um minicontrato é definida em R$ 0,20 por ponto do contrato futuro. Sendo que, para se negociar minicontratos o investidor precisa ter como garantia (ou margem de segurança) na conta de sua corretora o valor mínimo de 15% do valor total do minicontrato negociado. Todos os dias as posições em aberto (compradas ou vendidas) dos minicontratos futuros são ajustadas pela BM&F, creditando e debitando valores financeiros nas contas dos investidores. No dia de vencimento, a posição de cada investidor será automaticamente zerada. Por fim, destaca-se que existe incidência de imposto de renda (15 %) sobre o lucro líquido em operações de minicontratos futuros.

Para exemplificar o que foi destacado anteriormente, considere uma situação hipotética em que um investidor compra um minicontrato correspondente ao Ibovespa futuro (por exemplo, para o mês de fevereiro), que tem seu valor cotado em 59 300 pontos na abertura do pregão da BM&F. O investimento inicial desta transação será de 11 860 reais, ou seja, 20 % do valor do contrato cheio para o Ibovespa futuro de fevereiro. Se durante o fechamento no final do dia, o mesmo Ibovespa futuro fechou valendo 59 900 pontos, então o saldo em conta será de 11 980 reais. Se ao final do dia o investidor vende o minicontrato, então será creditado na conta do investidor, um crédito de 120 reais que corresponde a R$ 0,20 por cada ponto de valorização (600 pontos) do índice futuro no dia. Se caso ocorra o contrário e o índice futuro para fevereiro fechou o dia em 57 167 pontos, então será debitado um valor de 426,6 reais, correspondente a R$ 0,20 x 2133 pontos.

Com base no anterior, foi implementado uma estratégia de negociação de compra e venda de minicontratos, a qual teve como premissa fundamental a previsão sobre o

sentimento do mercado com relação a um dia de negociação, usando como base as notícias reportadas no dia anterior, ou seja, as notícias ao longo de um dia sugerem uma previsão para a tendência do mercado no dia seguinte. Desta forma, se a previsão for positiva, então pode-se comprar um minicontrato (ou contrato cheio) do índice (Ibovespa) na abertura do pregão e a partir desse momento têm-se diferentes opções de estratégia que dependem exclusivamente dos resultados subsequentes das previsões obtidas. A estratégia final adotada, para um melhor entendimento, pode ser visualizada na figura 6.9 e resumida da seguinte forma:

a) Suponha que o sistema implementado prevê que a expectativa da tendência do mercado é definida como positiva para um dia de negociação analisado, então compra-se na abertura.
b) Uma vez comprado -> Verifica-se previsão para o dia seguinte
   2.1 Se a previsão for positiva então mantém-se o ativo em custódia e retorna ao item (a).
   2.2 Se a previsão for negativa então vende-se na abertura
      2.2.1 Se no fechamento o ativo confirmou uma queda -> O ativo é recomprado ficando em custódia e retorna ao item (a).
      2.2.2 Se no fechamento o ativo não confirmou uma queda -> Mantém-se a venda (posição zerada) e retorna ao item (a).

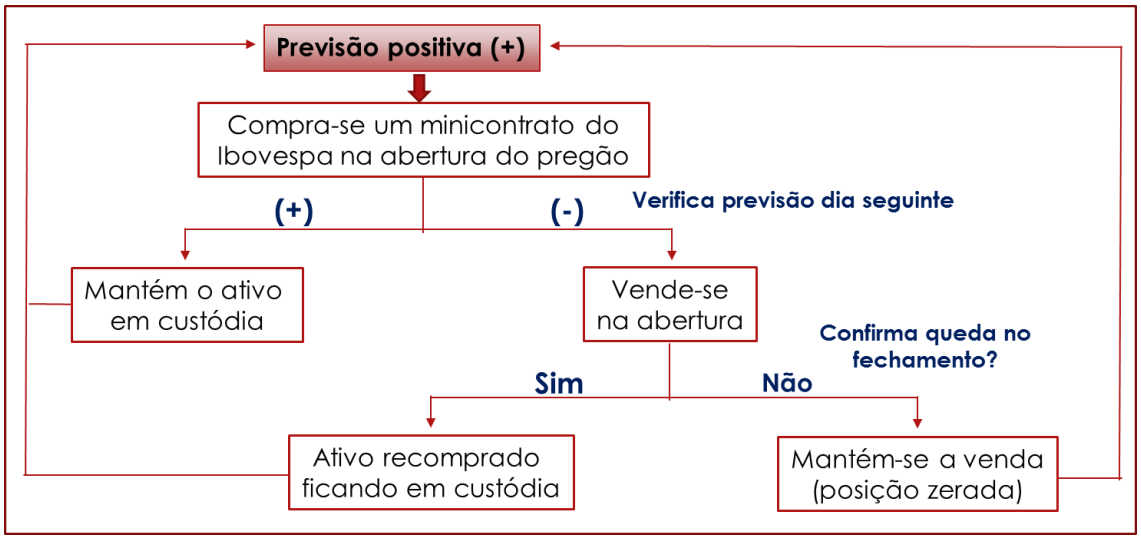

Figura 6.9 – Estratégia de negociação implementada

Perceba que a estratégia tenta manter ou seguir a tendência positiva caso exista uma sequência de verdadeiros positivos (TP). Já os verdadeiros negativos (TN) são incorporados (ou aproveitados) na estratégia mediante uma operação de *day-trade* (venda e compra no mesmo dia). No caso de falsos positivos (FP), mesmo assim a estratégia é mantida, enquanto que para falsos negativos (FN) a estratégia de curto prazo é anulada e recomeçada a partir do dia seguinte. Para detalhes dos valores TP, TN, FP e FN veja seção 7.4.

Para as recomendações de compras e vendas dos minicontratos (estratégia adotada anteriormente) foi levado em consideração à classificação final dos documentos obtida pelo classificador de maior acurácia no conjunto de teste utilizado. Valores como custos da transação na BmfBovespa, assim como a incidência de imposto de renda, não foram considerados no sistema de negociação proposto, dado que a definição destes custos é difícil de ser quantificado, pois os mesmos dependem de alguns fatores principalmente como a corretora envolvida, entre outros custos.

Em relação aos dados utilizados referentes aos valores dos minicontratos do Ibovespa no período simulado, os mesmos foram obtidos através do site ADVFN[17] e correspondeu ao período de negociação entre 07/03/2013 a 20/05/2013. Um gráfico mostrando as variações do minicontrato pode ser visualizado na figura 6.10.

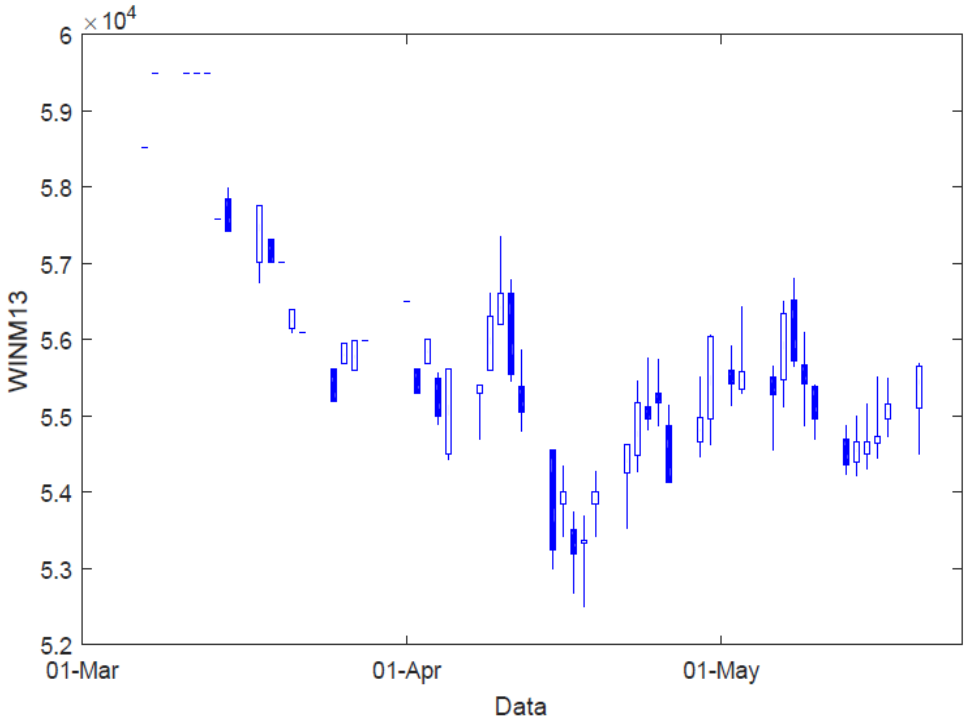

Figura 6.10 - Variação do minicontrato WINM13 durante os dias anteriores a seu vencimento em junho de 2013

[17] https://br.advfn.com/

Cabe destacar que este período analisado é relativamente curto quando comparado ao período total de previsão realizado neste trabalho (janeiro de 2012 a maio de 2013) e que isto ocorreu devido ao fato de estes foram os únicos dados disponíveis no site citado. Outras fontes na internet também foram consultadas, principalmente na tentativa de se obter dados referente ao ano de 2012, porém até o presente momento não foi possível obtê-los.

E por fim, deve-se salientar também que a justificativa das estratégias de negociação adotada estarem centradas na compra e venda de minicontratos do Ibovespa, enquanto o trabalho proposto visou a previsão do Ibovespa, é porque aceita-se a hipótese de que o minicontrato reproduz em alto grau a tendência do indicador base, ou seja, do Ibovespa.

7. Resultados e Análises

Através da metodologia adotada no capítulo anterior, vários modelos foram desenvolvidos a fim de se obter o objetivo principal desta tese que foi a previsão do índice do mercado acionário brasileiro (Ibovespa). Algoritmos e técnicas bem atuais, como a representação distribuída através do algoritmo *word2vec*, as redes neurais DLNNs, diferentes métricas de associação, entre outras, foram analisados e utilizados colaborando para um melhor desempenho dos modelos implementados. Nos experimentos realizados a arquitetura do modelo I-CNNELM proposto neste trabalho foi comparada com três outros modelos (I-CNN e I-ELM e I-Bok) e os resultados obtidos foram analisados e avaliados através das medidas de avaliação citadas no capítulo 6. Finalmente uma estratégia de negociação foi implementada para medir a lucratividade dos resultados obtidos pela metodologia proposta.

7.1. Base de dados: Séries temporais e notícias econômicas

A base de dados de séries temporais utilizadas nesta tese foi composta de cotações do fechamento diário do Ibovespa no mesmo período relacionado a base de dados nprev, ou seja, janeiro de 2012 a maio de 2013.

Inicialmente as séries temporais foram processadas de forma a calcular o valor das variações relativas do Ibovespa no período analisado. Estas variações estão relacionadas a tendência diária do Ibovespa. A partir dos valores das tendências calculados, o passo seguinte visou a discretização das mesmas conforme estratégia adotada na seção 6.1.3. Um gráfico contendo as cotações do Ibovespa juntamente com a variação relativa do mesmo no período analisado pode ser visualizado na figura 7.1, onde é possível visualizar que a série temporal utilizada possui variações bruscas, sem representar uma tendência definida ao longo do período. Esta característica das séries temporais pode ser um ponto importante para o trabalho desenvolvido, pois indica que os modelos desenvolvidos foram testados em condições complexas.

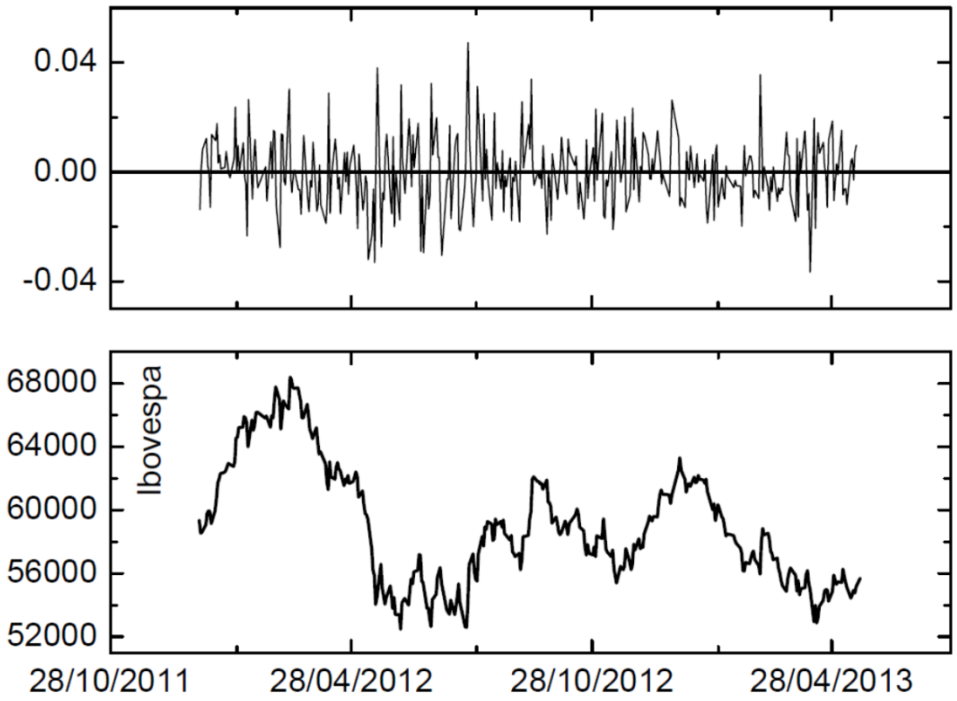

Figura 7.1 - Gráfico superior contendo variação relativa (%) do Ibovespa e gráfico inferior com cotações de fechamento do Ibovespa no período analisado

Em relação a base de dados textual (nw2v), a mesma consistiu de 380592 títulos de notícias recuperados a partir dos principais sites de notícias econômicas do Brasil no período de janeiro de 2012 a dezembro de 2016. Todos estes títulos foram utilizados para gerar a representação distribuída das palavras (vetores *word embeddings*) e tentar encontrar padrões de comportamentos nos mesmos através do algoritmo *word2vec*. A base de dados em questão é de domínio econômico e financeiro, ou seja, todas as notícias utilizadas, de alguma forma estão relacionadas com o mercado de ações brasileiro. Sendo assim, cabe ressaltar que muitos dos trabalhos envolvendo o uso do algoritmo *word2vec* na maioria das vezes fazem uso de vetores *word embeddings* pre-treinados com milhões de tokens e disponíveis publicamente. Entretanto, tem sido demonstrado que o desempenho dos modelos pode ser melhorado através do treinamento do *word2vec* usando dados específicos de domínio (HUGHES *et al*., 2017). Este foi um dos motivos que levou a autora a adotar a opção de se fazer o treinamento do *word2vec*, mesmo que com uma quantidade limitada de tokens. A dificuldade em se encontrar dados treinados confiáveis (ou vetores *word embeddings*) na língua portuguesa também contribuiu para esta decisão. Ademais, através de alguns resultados do treinamento do *word2vec* apresentados na seção 6.2.2 é possível verificar a qualidade dos vetores gerados com a base de dados utilizada.

Já em relação aos dados textuais (base de dados nprev) que foram utilizados na etapa de classificação ou previsão do Ibovespa optou-se por utilizar apenas uma pequena parte da

base de dados, isto é, apenas 162001 títulos de notícias conforme justificado na seção 6.1.1 A partir deste conjunto foram utilizadas algumas estratégias na definição de alguns parâmetros do modelo como por exemplo: o uso da tendência do Ibovespa ao invés de usar o próprio valor do índice, formas de atribuição de classes às notícias e quantidade de classes na qual foi dividida a base de dados. Todas estas estratégias tiveram como base o dia e a hora da publicação das notícias, que foi fator determinante para a conexão entre o mercado de ações e as notícias utilizadas.

7.2. Parâmetros do modelo I-CNNELM

A base de dados, assim como toda a metodologia utilizada para alcançar os objetivos deste trabalho foram apresentadas e detalhadas no capítulo anterior. Vários modelos foram testados com diferentes configurações de parâmetros buscando uma estrutura que apresentasse um melhor desempenho e consequentemente uma melhor capacidade de generalização. As entradas para os modelos implementados foram constituídas de vetores *word embeddings* gerados a partir da utilização do algoritmo *word2vec* em textos econômicos compostos de palavras ou *tokens* unigramas e trigramas. Sendo estes últimos obtidos automaticamente a partir dos textos analisados utilizando sete métricas de associação diferentes conforme em 6.2.1. Na tabela 7.1 pode ser visualizado um resumo dos principais parâmetros que foram alterados e testados ao longo do experimento realizado com o modelo I-CNNELM. A quantidade de camadas convolucionais da rede CNN, assim como a quantidade de camadas *de pooling* não foram alteradas, por isto não fazem parte da tabela apresentada. Seguindo o trabalho de (KIM, 2014) optou-se por utilizar apenas uma camada convolucional seguida por uma camada de *pooling* e uma única camada *fully connected*. Dois hiperparâmetros relacionados aos filtros da CNN também não foram alterados, ou seja, a utilização ou não do *padding* (optou-se pela utilização da forma *narrow convolution*, quando não há expansão da matriz) e o tamanho do *stride* que foi fixado em 1.

Os diferentes modelos obtidos a partir das possíveis configurações citadas foram testados e avaliados utilizando o método de validação cruzada (validação cruzada de *k* partições), onde o resultado final foi a média da acurácia nos *k* conjuntos utilizados para testar a performance dos modelos implementados. Uma análise sobre todos os resultados obtidos para os diferentes modelos gerados e validados são apresentados nas próximas seções.

Tabela 7.1 - Resumo dos principais parâmetros do modelo I-CNNELM

Parâmetros do pré-processamento dos textos e word2vec	
1 - métrica de associação	7 métricas
2 – filtro de frequência (ω)	50 e 100
3 - *size* ou dimensão do vetor *word embedding* (d)	100 e 300
Hiperparâmetros da CNN	
4 - quantidade de filtros convolucionais	32, 64, 96, 128 e 160
5 - tamanho dos filtros convolucionais	3, 5 e 7
Parâmetros da ELM	
6 - quantidade de neurônios da camada escondida da ELM	67, 135 e 200

7.3. Análise e avaliação dos parâmetros do modelo I-CNNELM

Seguindo os procedimentos e parâmetros descritos acima foram testados aproximadamente 1260 modelos de previsão da seguinte forma. A partir do pré-processamento dos textos da base de dados nw2v, para as 7 diferentes métricas utilizadas, dois parâmetros de filtragem da frequência (ω = 50 e 100) e duas dimensões para os vetores *word embeddings* (d =100 e 300) foram gerados 28 conjuntos destes vetores representando cada um, uma possível configuração para a base de textos utilizada. Para cada uma destas 28 configurações obtidas variando os diferentes parâmetros do pré-processamento (ou seja, métricas, ω e d) diferentes arquiteturas de redes neurais foram testadas a fim de encontrar uma acurácia máxima para cada conjunto dos parâmetros utilizados no processamento dos textos. Sendo assim, as redes CNN foram configuradas com uma camada convolucional, uma camada de *pooling* e uma camada *fully connected*, *stride* igual a 1 e não utilização do *padding* (*narrow convolution*). Por sua vez, a quantidade de filtros convolucionais variou no intervalo de 32 a 160 (com passo de 32), enquanto os tamanhos destes filtros foram definidos como 3, 5 e 7. Por outro lado, para cada configuração da CNN, outras configurações da rede ELM foram testadas, onde o número de neurônios da camada escondida foi definido como 67, 135 e 200.

Conforme mencionado anteriormente, o treinamento e previsão dos modelos I-CNNELM tendo diferentes configurações, foram executadas utilizando a base de dados composta por notícias e tendências do Ibovespa no período de janeiro de 2012 a maio de 2013. Por fim, cabe ressaltar então que para cada configuração utilizada no processamento dos textos nw2v (28 no total) foi encontrado uma arquitetura de rede neural ótima (CNN+ELM) que forneceu um valor de acurácia máxima para as previsões do Ibovespa. Estes resultados podem ser visualizados na tabela 7.2 apresentada a seguir.

Tabela 7.2 - Resultados da acurácia obtidos com diferentes configurações

Medidas de Associação	**ω = 50 d = 100**	**ω = 50 d = 300**	**ω = 100 d = 100**	**ω = 100 d = 300**
Informação Mútua Pontual	54,90	57,86	58,14	**58,51**
T-Student	54,95	56,08	**56,09**	54,29
Chi-quadrada	57,29	56,35	**59,05**	56,45
Razão de Verossimilhança	**57,93**	56,38	56,39	57,26
Mi-like	54,05	56,98	56,61	**60,22**
Poison-Stirling	54,64	55,17	**58,16**	57,61
Jaccard	55,81	54,96	**57,27**	56,39

7.3.1. Pré-processamento dos textos e parâmetros do *word2vec*

Baseado na tabela apresentada (7.2) é possível verificar que os resultados encontrados para todas as métricas utilizadas não apresentaram valores de acurácia muito diferentes quando comparadas entre si, principalmente se forem analisadas apenas os modelos de maior acurácia para cada métrica de associação. Apesar de algumas métricas terem a tendência de atribuir altas pontuações para itens com maior frequência (como por exemplo, Razão de Verossimilhança e *T-Student*) enquanto outras privilegiam baixas frequências, os trigramas encontrados foram semelhantes porém ranqueados de formas diferentes. A fim de se utilizar um número grande de trigramas, optou-se neste trabalho por utilizar os 90% dos trigramas mais bem ranqueados, descartando apenas trigramas com pontuação zero ou com valores

muito pequenos. Isto colaborou para que quase todos os trigramas encontrados em todas as possíveis métricas fossem utilizados no processamento. Consequente, os vetores *word embeddings* gerados para todos os textos modificados também foram semelhantes. Acredita-se então que este fato pode ser uma possível justificativa para a pouca variação nos resultados encontrados. Ademais, através dos resultados apresentados e dos valores destacados (os 7 modelos de maior acurácia) é possível verificar que apenas a métrica T-Student não apresentou acurácia maior que 57%. As duas métricas com maior acurácia foram a Chi-quadrada (59,05%) e a Mi-like (60,22%).

Outro resultado importante a ser destacado está relacionado ao parâmetro de filtragem de frequência (ω). Analisando a tabela é possível verificar que utilizando ω =100 há uma ligeira melhora nos resultados obtidos para algumas das métricas utilizadas. De fato, 5 das maiores acurácias foram obtidas com esta configuração. Isto demonstra que um valor de ω adequado pode ajudar a identificar as palavras mais relevantes nos textos e contribuir para reduzir a dimensionalidade dos tokens ou palavras-chave a serem processados. Já em relação à dimensão dos vetores *word embeddings* (d), onde cada token foi representado por um vetor de d-dimensão, para a maioria dos resultados de maior acurácia foi encontrado d=100. Entretanto o melhor modelo apontado na tabela (acurácia de 60,22%) tem configuração de d=300, o que está em linha com os resultados de melhor acurácia reportados no trabalho de (MIKOLOV *et al*., 2013a).

7.3.2. Hiperparâmetros da CNN

Os resultados reportados na tabela 7.2, de uma maneira geral, dependem dos parâmetros das arquiteturas das redes neurais CNN e ELM. Dentre as possíveis configurações utilizadas nesta tese, dois parâmetros foram avaliados para a arquitetura de rede CNN (quantidade e tamanho dos filtros convolucionais). Na figura 7.2 é possível visualizar o comportamento destes parâmetros para os sete melhores modelos (maior acurácia) reportados na tabela 7.2.

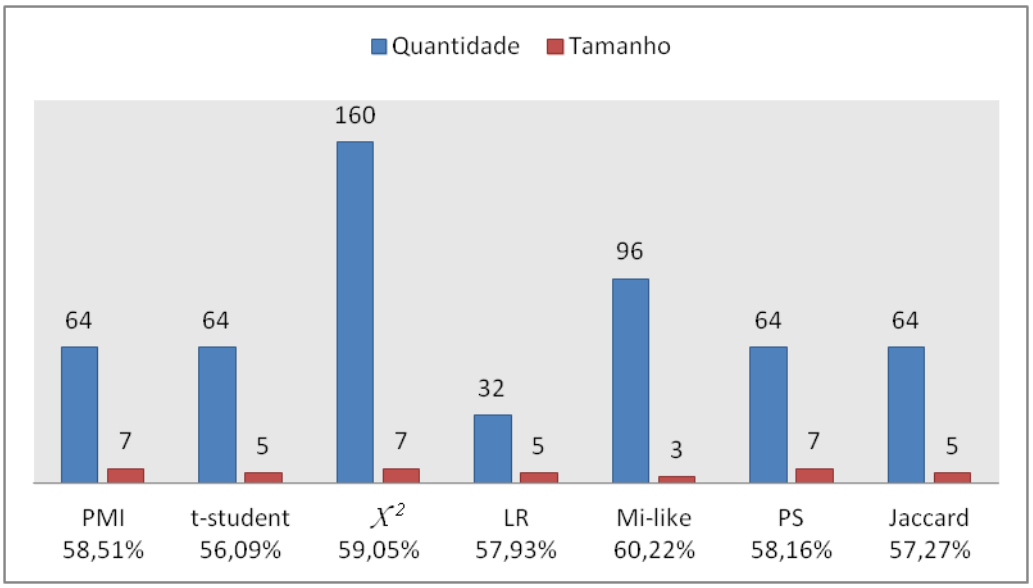

Figura 7.2 - Quantidade e tamanho dos filtros convolucionais para os sete modelos de maior acurácia

Na figura apresentada, PMI corresponde a métrica Informação Mútua Pontual, X^2 é Chi-quadrada, LR está associada à Razão de Verossimilhança e PS é *Poison-Stirling*. Através do gráfico apresentado foi possível verificar que os melhores resultados para as diferentes configurações apresentadas foram obtidos utilizando uma quantidade maior de filtros convolucionais e também um tamanho (ou kernel) maior. A maioria dos resultados apontados fizeram uso de 64 filtros convolucionais com kernel igual a 5 e 7. Isto implica em afirmar que para se obter características relevantes a partir de dados altamente complexos, como é o caso dos dados utilizados nesta tese, é necessária uma quantidade maior de filtros convolucionais. No entanto, em relação ao tamanho do filtro convolucional é possível destacar que apesar da maioria dos resultados terem valores 5 e 7, para o melhor resultado (acurácia de 60,22%) o tamanho do filtro foi definido como 3, mais precisamente 3 x 300 levando em consideração a dimensão do vetor *sentence embedding* (*d* igual a 300) utilizado nesta configuração.

7.3.3. Parâmetro da ELM

A facilidade de configuração das redes ELMs é uma de suas principais características a ser destacada. A quantidade de neurônios da camada escondida e a função de ativação são os dois principais parâmetros a serem ajustados neste modelo de RNA. Sendo assim, nesta tese optou-se pela modificação apenas da quantidade de neurônios da camada escondida que

foi definida segundo as regras apresentadas na seção 6.3, mais especificamente os valores utilizados e modificados foram 67, 135 e 200.

Na figura 7.3 pode ser visualizado como se comportaram os resultados dos melhores modelos apresentados na tabela 7.2 e correspondente às diferentes configurações utilizadas nesta tese, quando analisado o parâmetro relacionado à quantidade de neurônios da camada escondida da ELM.

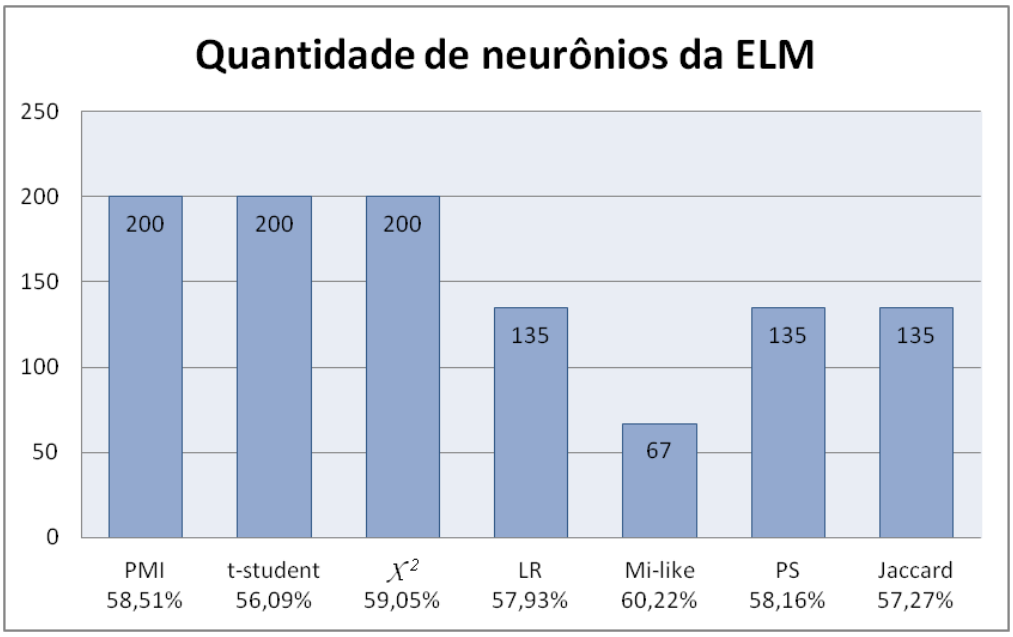

Figura 7.3 - Quantidade de neurônios da camada escondida da ELM para os sete modelos de maior acurácia

Na figura apresentada é possível verificar que para a maioria dos modelos citados é necessária uma quantidade maior de neurônios para encontrar o relacionamento entre o conteúdo dos títulos das notícias (representadas por vetores *sentence embeddings*) e a tendência dos valores do fechamento do Ibovespa. Todavia, o fato de simplesmente adicionar mais neurônios na camada escondida não garante a melhor performance dentro das possíveis configurações sugeridas nesta tese. Isto pode ser constatado por meio do resultado do modelo de maior acurácia (60,22%) encontrado, onde foram necessários apenas 67 neurônios. Esta característica também foi constatada no parâmetro relacionado ao tamanho do filtro convolucional citado anteriormente.

7.4. Análise e avaliação do desempenho do modelo I-CNNELM

Todos os parâmetros citados nas seções anteriores foram fundamentais para a definição de uma arquitetura adequada e eficiente para atingir os objetivos propostos neste trabalho. O melhor resultado encontrado no modelo de previsão proposto dentre todas as possíveis configurações testadas obteve um valor de acurácia de 60,22%, com a métrica Mi-like e parâmetros de pré-processamento dos textos ω=100 e d=300, incluindo uma CNN com 96 filtros convolucionais (com tamanho 3) e uma rede ELM de 67 neurônios escondidos.

Os resultados obtidos pelos modelos de classificação podem ser representados por uma estrutura denominada matriz confusão. É importante o uso desta estrutura, pois a mesma permite uma análise mais eficiente do desempenho do classificador analisando os resultados de cada classe individualmente, ou seja, é possível verificar quando a classe é positiva, mas o classificador diz que é negativa ou vice-versa. Este conhecimento é importante porque o fato de um classificador ter tendência a errar uma classe ou a outra pode impactar e muito nos resultados obtidos através das decisões tomadas por meio da análise desta estrutura. Isto, por exemplo, pode ser muito importante para construir estratégias de negociação de compra e venda no mercado de ações dando pesos diferentes às decisões de negociação, dependendo se a classe prevista tende a ser positiva ou negativa.

Na tabela 7.3 pode ser visualizada a matriz confusão relacionada ao modelo de maior acurácia (60,22%), onde os valores apresentados na diagonal da tabela TP (verdadeiro positivo) e TN (verdadeiro negativo) representam as tendências do Ibovespa positivas e negativas classificadas corretamente e os valores fora da diagonal correspondem às tendências do Ibovespa classificadas erroneamente, isto é, FN corresponde ao falso negativo (tendência positiva mas prevista como negativa) e FP corresponde ao falso positivo (tendência negativa porém prevista como positiva). Sendo assim, se os elementos da diagonal forem somados e divididos pelo número total de eventos em cada célula, tem-se o valor da acurácia do modelo, que é de 60,2%.

Ademais, a qualidade do modelo híbrido proposto também pode ser quantificada por outros indicadores derivados a partir da tabela 7.3. Considerando a classe positiva, existem dois parâmetros que ajudam a medir as características do classificador. A primeira está relacionada a revocação (*recall*, *true positive rate* ou *sensitivity*), na qual quantifica as taxas

de previsões positivas reais, ou seja, previsões positivas (TP) divididas pelo número total de tendências positivas reais do modelo (TP+FN), totalizando 59,6%. Por outro lado, o outro indicador, a precisão (*precision* ou *positive predictive value*) está relacionada a taxa de acerto quando um modelo prevê uma tendência positiva, isto é, o número de previsões positivas corretas (TP) dividido pelo número total de previsões positivas (TP + FP), cujo valor para o modelo analisado foi de 56,7%. Dentro deste contexto, o parâmetro denominado Medida-F (F-score) quantifica uma média harmônica entre a revocação e a precisão. Quanto mais próximo de 1 é o valor desta medida, melhor será o classificador, sendo a mesma utilizada eventualmente com um bom indicador da acurácia do modelo preditivo. Para o modelo analisado o valor desta medida foi de 58%.

Tabela 7.3 - Matriz confusão do modelo de previsão de maior acurácia

Classes		**Prevista**		
		Positiva	**Negativa**	
Real	**Positiva**	**TP = 93**	**FN = 63**	**156**
	Negativa	**FP = 71**	**TN = 110**	**181**
		164	**173**	**Total = 337**

Além disto, a partir da tabela 7.3 é possível definir também uma medida de especificidade (*true negativo rate*) para classes negativas. Este parâmetro quantifica a proporção de tendências negativas reais corretamente previstas (TN) e pode ser calculado como TN/TN+FP, o que resulta em um valor de 60,8% para o modelo analisado. Deste modo, analisando os valores obtidos a partir da tabela 7.3 para as medidas revocação e especificidade, os mesmos sugerem que o modelo analisado é quase estatisticamente equivalente em evitar resultados falsos para ambas as classes (tendências positivas e negativas). O equivalente do parâmetro precisão, porém para a classe negativa é o *negative predicted value* calculado por TN/TN+FN, cujo resultado foi de 63,6%. Por fim a medida F para a classe negativa foi de 62%. Este valor é ligeiramente maior que a medida F da classe positiva (58%) e pode indicar que o modelo é ligeiramente mais eficiente em previsões relacionadas à classe negativa. Este resultado pode estar associado ao fato de que a quantidade de amostras com classes negativas (181) utilizadas no treinamento do modelo é

superior a quantidade de amostras com classes positivas (156), levando ao algoritmo de treinamento a reconhecer melhor as amostras negativas.

Por fim, cabe ressaltar que conforme citado ao longo deste capítulo o melhor resultado encontrado dentre todas as possíveis configurações testadas obteve um valor de acurácia de 60,22%. Este resultado está em linha com alguns resultados reportados na literatura e descritos no capítulo 5, porém envolvendo outros mercados financeiros, onde é possível destacar que o mercado mais analisado é o mercado americano, como por exemplo, o S&P500.

7.5. Comparação entre resultados

Uma das propostas deste trabalho foi a comparação entre os resultados obtidos nesta tese através da implementação do modelo I-CNNELM e os resultados obtidos utilizando três outros modelos (I-CNN, I-ELM e I-Bok) que também foram implementados e analisados aqui conforme detalhado na seção 6.4. A ideia principal foi verificar se modelos de previsão (neste caso em específico, previsão do Ibovespa) baseados na combinação das arquiteturas de redes neurais CNN + ELM e do método de representação de documentos através dos vetores *sentence embeddings* poderiam apresentar resultados superiores quando comparados a: (i) modelos de previsão onde as arquiteturas CNN e ELM foram utilizadas separadamente e a técnica de representação das palavras foram respectivamente, vetores *sentence embeddings* e *bag-of-words* e (ii) modelo de previsão (I-Bok) com arquitetura de rede MLP com 4 camadas escondidas e técnica *bag-of-words* para representação das palavras.

Os resultados reportados na tabela 7.4 correspondem a acurácia do modelo proposto I-CNNELM e dos 3 modelos utilizados para a comparação. Cabe ressaltar que a configuração dos parâmetros dos modelos I-CNN e I-ELM e I-Bok para os resultados apresentados nesta tabela estão em concordância com os parâmetros que fizeram parte da configuração dos cinco modelos de maior acurácia do modelo híbrido proposto (I-CNNELM). De uma maneira mais detalhada, tomando como exemplo a arquitetura com melhor desempenho na tabela 7.2, a configuração de parâmetros da mesma foi separada em três modelos. O primeiro contendo uma etapa de processamento de texto (Mi-like, ω=100 e d=300) conectada a uma rede de arquitetura CNN com 96 filtros convolucionais de tamanho 3xd (I-CNN). E o segundo possui os mesmos parâmetros do processamento de texto, com a representação de palavras *bag-of-words*, mas agora conectada a uma rede neural ELM com 67 neurônios na camada oculta (I-

ELM). E por fim, o terceiro modelo contendo também os mesmos parâmetros do processamento de texto, com a representação de palavras *bag-of-words* e conectada a uma rede neural MLP de 4 camadas escondidas com 1024 neurônios para cada uma (I-Bok).

Tabela 7.4 - Comparação entre modelos. Mesma configuração de parâmetros dos 5 modelos I-CNNELM de maior acurácia (%)

I-CNNELM	I-CNN	I-ELM	I-Bok
60,2	50,8	40,0	52,0
59,0	49,0	40,0	51,0
58,5	51,9	47,0	50,0
58,1	51,6	40,0	52,0
57,9	51,9	46,0	52,0

Através da tabela é possível observar que o modelo híbrido proposto supera os resultados obtidos com os demais modelos testados. Porém, dada a grande quantidade de arquiteturas simuladas com o modelo I-CNNELM, surgiu uma dúvida referente a seguinte questão, qual seria a possibilidade de uma melhora nos resultados obtidos pelos modelos utilizados na comparação (I-CNN, I-ELM e I-Bok), caso os mesmos fossem testados também sob todas as possíveis configurações assumidas pelo modelo proposto e não somente pelas configurações dos modelos de maior acurácia. Para esclarecer melhor este ponto, foram então simulados e testados 420, 84 e 28 modelos para cada arquiteturas de redes neurais respectivamente (I-CNN, I-ELM e I-Bok), seguindo sempre as mesmas configurações usada no modelo híbrido.

Sendo assim, considere por exemplo, as configurações específicas utilizadas no processamento dos textos, ω = 50 e ω = 100 com d = 100 e d = 300. Para cada combinação específica dos parâmetros ω e d, as mesmas arquiteturas da rede CNN que fizeram parte do modelo hibrido foram testadas, ou seja, arquiteturas CNN com filtros variando entre 32 e 160 (passo = 32) e com tamanhos 3, 5 e 7. De uma maneira mais detalhada pode-se dizer que para uma combinação específica de parâmetros ω e d, no total foram testadas 15 configurações de redes CNN. Por fim, para cada arquitetura testada (ω, d, número de filtros e tamanho de filtro) a acurácia foi calculada e as cinco maiores foram utilizadas na comparação com os melhores resultados do modelo hibrido.

Esta mesma abordagem foi utilizada para a arquitetura ELM. Por exemplo, para a mesma configuração dos parâmetros utilizados no processamento de textos (ω, d), a mesma

configuração que faz parte da rede ELM do modelo hibrido, a saber, quantidade de neurônios da camada escondida assumindo os valores de 67, 135 e 200, foram testadas. Assim como no modelo citado anteriormente, para uma combinação específica de parâmetros ω, *d*, no total foram simuladas 3 arquiteturas de redes ELM e os resultados com maiores acurácias foram comparadas ao modelo hibrido. O mesmo é válido para o modelo I-Bok, porém utilizando a arquitetura de rede neural MLP, isto é, mesma configuração de parâmetros para o processamento dos textos, porém com uma arquitetura de rede MLP com 1024 neurônios na camada escondida.

Finalmente uma tabela com os modelos de maior acurácia para os três modelos citados foi gerada, onde novamente constatou-se claramente a superioridade do modelo híbrido. Estes resultados podem ser visualizados na tabela 7.5.

Tabela 7.5 - Comparação entre modelos. As cinco configurações de maior acurácia para todos os modelos dentre todas as possíveis configurações utilizadas

Modelos	**Acurácia (%)**				
I-CNNELM	60,2	59,0	58,5	58,1	57,9
I-CNN	57,0	57,0	56,0	56,0	55,0
I-ELM	55,0	51,0	49,0	47,0	47,0
I-Bok	56,0	55,0	54,0	53,0	53,0

A partir da tabela algumas observações podem ser destacadas:

- A primeira delas está relacionada a superioridade do modelo proposto nesta tese. É possível observar que nenhum dos modelos utilizados na comparação alcançou a acurácia máxima obtida (60,2%). Outro ponto a ser destacado é que dentre as cinco melhores configurações relacionados ao modelo hibrido (I-CNNELM), o valor de menor acurácia foi de 57,9% e que dentre todas as possíveis configurações dos modelos de comparação, nenhum deles conseguiu alcançar este valor de acurácia.
- Analisando somente os três modelos utilizados na comparação é possível afirmar que o melhor resultado foi obtido utilizando a arquitetura de rede CNN e representação *word embedding* (I-CNN) com acurácia de 57,0%. O pior resultado foi encontrado utilizando apenas a rede ELM (I-ELM, 47,0%) com representação *bag-of-words*. Isto sugere que a combinação destas duas arquiteturas de redes juntamente da

representação *word embedding* é melhor, pois resulta em uma performance mais competitiva conseguindo extrair conhecimento relevante a partir de dados textuais.

- Fazendo uma análise mais restrita e comparando apenas os modelos que fazem uso da técnica *bag-of-words* (I-ELM e I-Bok) pode-se observar a superioridade das redes MLPs que fez parte do modelo I-Bok. Acredita-se que esta superioridade pode estar relacionada ao grande número de camadas escondidas envolvidas na configuração deste modelo (4 camadas ocultas de 1024 neurônios cada).

Finalmente, pode-se destacar também o tempo de execução que foi utilizado para cada modelo. Na literatura destaca-se muito o uso de placas GPUs para o processamento de dados textuais ou imagens a partir de arquiteturas DLNN. De fato, quanto maior o conjunto de dados a ser analisado e mais complexa a arquitetura DLNN, maior será o tempo de execução das tarefas, seja classificação ou regressão. De uma forma particular, o treinamento de uma rede CNN por exemplo, pode ser bastante lento devido à quantidade de cálculos necessários para cada iteração. O uso de gpus pode ajudar a acelerar este cálculo. Entretanto, máquinas dotadas de várias gpus ainda não são realidade em todos os institutos de pesquisas voltados para Inteligência Artificial ou mesmo em empresas que fazem uso deste tipo de tecnologia. Neste trabalho em específico não foi possível fazer o uso de GPUs para o processamento dos modelos testados. Todas as simulações foram realizadas utilizando o software *Python* 3.5 e sendo executado em um PC com 3.4 GHZ, CPU com 32 cores e 256 GB de Ram. Na tabela 7.6 apresentada é possível visualizar o tempo de execução para os modelos apresentados na tabela 7.5 citada anteriormente e seus respectivos tempos médio de processamento na etapa de treinamento na máquina utilizada.

Tabela 7.6 - Tempo médio de treinamento dos 5 modelos de maior acurácia de cada arquitetura analisada

Modelos	**Acurácia (%)**					**Tempo médio de treinamento**
I-CNNELM	60,2	59,0	58,5	58,1	57,9	1h41min20seg
I-CNN	57,0	57,0	56,0	56,0	55,0	27hs45min26seg
I-ELM	55,0	51,0	49,0	47,0	47,0	1min30seg
I-Bok	56,0	55,0	54,0	53,0	53,0	34min55seg

Analisando os tempos médios reportados é possível constatar que se for considerado uma relação entre acurácia e o tempo de processamento, o modelo proposto nesta tese (I-

CNNELM) é o que possui um melhor custo benefício, pois além de apresentar os maiores valores de acurácia, também apresentaram um tempo de processamento bem inferior quando comparados as redes CNN (I-CNN considerado como o segundo melhor modelo analisado). Pode-se observar também que o uso somente das redes ELM, através do modelo I-ELM apresentou um tempo bem inferior quando comparados aos demais modelos. Entretanto, se for analisado a acurácia obtida por este modelo é possível constatar que os valores também são bem inferiores quando comparados principalmente com o modelo hibrido proposto. E por fim, cabe destacar que o uso das redes CNN através da configuração do modelo I-CNN foi o modelo que apresentou o maior tempo de processamento. De fato, muitos trabalhos envolvendo o uso das redes CNNs podem demorar dias de processamento, mesmo com o uso de gpus. E em aplicações envolvendo os mercados financeiros mundiais onde tudo é muito dinâmico e complexo, o uso de modelos onde o processamento dos dados possa ser realizado de uma forma eficiente e rápida pode significar um diferencial nas tomadas de decisões dos investidores. Dentro deste sentido pode-se citar a importância da metodologia citada nesta tese, onde se buscou o equilíbrio entre acurácia, eficiência e tempo de processamento.

7.6. Resultados da estratégia de negociação

Outro ponto abordado nesta tese foi a implementação de uma estratégia de negociação visando quantificar a lucratividade do modelo de maior desempenho (acurácia de 60,2%) obtido nos experimentos executados. A estratégia foi aplicada nos dados apresentados na figura 6.10. Entretanto, vale destacar que como o contrato tem um tempo hábil que é dado pelo vencimento, então inicialmente o mesmo é pouco negociado e à medida em que a data de vencimento se aproxima, o número de negócios aumenta. Por este motivo, foi considerado como primeira data para executar a estratégia, aquela onde o número de negócios superou a dezena pela primeira vez, o que correspondeu ao dia 04 de abril de 2013. A partir dessa data a estratégia de negociação foi implementada conforme discutido na seção 6.5 até a data de vencimento onde a posição foi necessariamente fechada.

Foram analisados 32 dias de negociação e o resultado final foi um lucro bruto correspondente a 1645 pontos. Isto equivale a um lucro líquido de 329 reais no período analisado. Observe que em uma estratégia de *buy and hold*, onde compra-se o ativo e o mantém até o final do vencimento, o lucro bruto seria de apenas 150 pontos (R$30,00), o que equivale a um retorno bem menor que o obtido através da metodologia proposta.

Finalmente, deve se destacar que esta situação analisada aqui, representa só um exemplo e que para se avaliar efetivamente o algoritmo do ponto de vista de estratégias financeira, outros testes podem ser necessários.

8.Conclusões

Nesta tese, foi apresentada uma combinação de arquiteturas de redes neurais artificiais (CNN + ELM) visando entender e predizer o comportamento dinâmico e não linear do mercado financeiro brasileiro (Bm&FBovespa). O modelo proposto emprega uma camada convolucional com diferentes filtros, alternada com uma camada de *pooling*, de forma a extrair características importantes e de mais alto nível a partir de dados de notícias econômicas obtidas nos principais portais de notícias econômicas do Brasil. Baseado nestas características, o classificador ELM no final da cadeia de processamento da rede CNN busca transformar as mesmas em conhecimento relevante para ajudar os investidores em suas tomadas de decisão, fornecendo uma solução determinística dos pesos de saída da rede. Diferentemente dos modelos que fazem uso das redes CNNs convencionais ou de outra arquitetura DLNN, não há iterações no processo de treinamento do modelo proposto, o que torna o mesmo muito mais rápido.

Os dados de notícias utilizados inicialmente passaram por uma etapa de pré-processamento, cujo objetivo foi tornar possível o uso dos mesmos pelos algoritmos de classificação propostos. Esta etapa constou de um processo de tokenização, limpeza dos dados e eliminação de *stopwords* e definição de n-gramas. Em seguida, os dados textuais processados foram transformados em representação numérica (vetores *word embeddings*), cuja metodologia utilizada foi o uso da representação distribuída obtida através do algoritmo *word2vec*. Finalmente, estes vetores foram utilizados como entrada (*input*) para os modelos de redes neurais utilizados.

Vários experimentos foram realizados, onde diferentes configurações de parâmetros foram analisadas e explicitadas. Através dos resultados obtidos nestes experimentos foi possível concluir que:

- Uma análise detalhada dos parâmetros analisados sinaliza que a melhor estrutura pode estar associada aos parâmetros utilizados no processamento dos textos, em particular com a dimensionalidade do espaço vetorial incluído no algoritmo *word2vec*.
- O uso de diferentes métricas de associação não acrescentou muita variação nos resultados obtidos, principalmente se forem analisados apenas os resultados para as configurações dos 7 modelos de maior acurácia. A métrica que fez parte do modelo de maior acurácia (60,22%) foi a Mi-like.

- A métrica de relacionamento utilizada para medir a qualidade dos vetores *word embeddings* obtidos no pré-processamento dos textos através do uso algoritmo *word2vec* mostrou uma forte correlação entre as palavras similares analisadas, comparável a pontuações sugeridas por especialistas humanos. Entretanto, como o algoritmo *word2vec* foi treinado com um número reduzido de tokens, não foi possível definir se um treinamento envolvendo uma quantidade maior de tokens (conforme sugerido em alguns trabalhos disponíveis na literatura) poderiam modificar os resultados finais obtidos.
- Em relação aos hiperparâmetros utilizados na configuração da rede CNN (quantidade de filtros convolucionais e tamanhos dos mesmos) é possível afirmar que uma quantidade maior de filtros pode resultar em modelos com maior acurácia. Entretanto, para o tamanho destes filtros (kernel) não foi possível chegar a uma conclusão específica, dado que a maioria dos resultados dos modelos de maior acurácia fizeram uso de kernel igual a 5 e 7, mas o modelo de maior acurácia (60,2%) precisou apenas de filtros com kernel igual a 3.
- Para a configuração da rede ELM, se for levado em consideração a maioria dos resultados dos sete modelos de maior acurácia, pode-se concluir que um número maior de neurônios na camada escondida da rede implica em uma melhor performance dos modelos. Entretanto, isto não se aplica se for considerado apenas o modelo de maior acurácia (60,2%).
- Os sete melhores resultados encontrados no modelo de previsão proposto, dentre todas as possíveis configurações testadas obteve valores de acurácia próximos de 60%, sendo o valor máximo obtido de 60,2%. Este desempenho foi semelhante a outros reportados na literatura, porém envolvendo mercados bem desenvolvidos e outras fontes de notícias.
- Uma análise da medida F (tanto para classe positiva quanto para a classe negativa) obtida a partir da matriz confusão para o modelo de maior acurácia (60,2%), indica que o modelo é ligeiramente mais eficiente em fazer previsões relacionadas a classe negativa.
- Através dos resultados obtidos nos experimentos realizados na comparação entre o modelo hibrido e outras três arquiteturas de redes neurais diferentes exploradas neste trabalho, constatou-se a superioridade da metodologia proposta. Isto sugere que a abstração de características exploradas através das

camadas convolucionais quando utilizadas junto com o classificador ELM é mais eficiente do que quando exploradas individualmente (CNN ou ELM). Ademais, treinar o modelo proposto é muito mais rápido do que outros métodos de redes DLNNs, em especial as redes CNNs.

- A estratégia de negociação adotada para verificar a lucratividade do modelo, sugere que é possível estudar o mercado de ações brasileiro através do seu principal indicador e obter uma rentabilidade considerável em negociações de compra e venda de seus minicontratos.

Em resumo, este trabalho forneceu novas perspectivas sobre o uso do processamento de dados textuais para prever o comportamento dos mercados financeiros, mas, neste caso, refere-se a um mercado emergente que, em geral, é menos consolidado. É possível destacar também que o método proposto é capaz de fornecer uma ferramenta automática e eficaz que pode beneficiar o investidor financeiro em suas tomadas de decisão ou mesmo em outras tarefas de classificação associadas ao uso de dados não estruturados, principalmente considerando textos na língua portuguesa, como por exemplo na classificação de textos envolvendo diagnósticos médicos. Deve-se ressaltar também que esses resultados preditivos foram alcançados sem o uso de dados estruturados, como preços históricos, etc., apoiando assim a ideia de que informações úteis podem ser recuperadas a partir de dados textuais, pelo menos para o sistema e as condições exploradas nesta tese.

8.1. Trabalhos futuros

Sistemas baseados em análise de notícias e que possam auxiliar o investidor em suas estratégias de negociações ainda é um campo pouco explorado na literatura, devido principalmente à dificuldade de se encontrar dados confiáveis disponíveis, assim como a complexidade inerente que envolve os estudos destes sistemas. Em relação ao sistema abordado, ainda existem muitos aspectos que podem ser considerados e investigados no futuro. O primeiro destes aspectos está relacionado na utilização da metodologia citada aqui, porém utilizando outras bases de dados de notícias e séries temporais do indicador Ibovespa (diferentemente da base nprev), como por exemplo, uma base de dados de 2015 a 2017, que foi um período conturbado na política do Brasil. O mesmo é válido para a estratégia de negociação, onde um período de tempo maior poderia ser considerado, dado a disponibilidade de dados para o estudo.

Outra questão ainda a ser discutida é qual deve ser o tempo que o mercado leva para reagir ao conteúdo de uma determinada notícia. Neste sentido, poderia ser interessante o uso de outras janelas de tempo de previsão, como por exemplo, previsões em um curto período de tempo (*intraday*).

O uso de novas variáveis, como indicadores macroeconômicos também poderiam ser investigados juntamente com dados textuais e a metodologia proposta aqui, de forma a melhorar a acurácia obtida e aumentar a lucratividade do sistema analisado.

Outros aspectos também podem ser avaliados como um estudo mais aprofundado na abordagem de armazenamento de dados da ordem de *Big Data*, assim como o estudo de outras arquiteturas computacionais para melhorar o desempenho do modelo proposto.

Por fim, muitos outros aspectos podem ser agregados à metodologia proposta, entretanto cabe destacar que os valores de acurácia encontrados, assim como a lucratividade do sistema proposto podem ser considerados animadores e sugerem novas investigações.

9.Referências

ACKLEY, D. H., HINTON, G. E. AND SEJNOWSKI, T. J. “A learning algorithm for Boltzmann machines.” **Cognitive science** 9.1 (1985): 147-169.

AHMED, A., YU, K., XU, W., GONG, Y., & XING, E. “Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks”. **In European Conference on Computer Vision** (pp. 69-82). Springer, Berlin, Heidelberg, 2008.

ARLOT, S., CELISSE, A. “A survey of cross-validation procedures for model selection”. **Statistics surveys**, 4, 40-79, 2010.

BALDI, PIERRE; SADOWSKI, PETER J. “Understanding dropout”. In: **Advances in neural information processing systems**. 2013. p. 2814-2822.

BALDI PIERRE; SADOWSKI, PETER. “The dropout learning algorithm”. **Artificial intelligence**, v. 210, p. 78-122, 2014.

BANERJEE, S., PEDERSEN, T. “The design, implementation, and use of the ngram statistics package”. **In International Conference on Intelligent Text Processing and Computational Linguistics** (pp. 370-381). Springer, Berlin, Heidelberg, 2003.

BAO, W., YUE, J., & RAO, Y. “A deep learning framework for financial time series using stacked autoencoders and long-short term memory”. **PloS one**, 12(7), e0180944, (2017).

BARONI, M., DINU, G., AND KRUSZEWSKI, G. “Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors”. **In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics** (Volume 1: Long Papers) (Vol. 1, pp. 238-247).

BENGIO Y, DUCHARME, R., VINCENT, P., JAUVIN, C. “A neural probabilistic language model”. **The Journal of Machine Learning Research** v.3, Feb (2003): 1137-1155.

BENGIO, Y., LECUN, Y. “Scaling Learning Algorithms Towards AI”. **Large-scale kernel machines**, v.34, n.5, p.1-41, 2007.

BENGIO, Y., LAMBLIN, P., POPOVICI, D., LAROCHELLE, H. “Greedy layer-wise training of deep networks”. **Advances in neural information processing systems**, pp. 153-160, MIT Press, 2007.

BENGIO, YOSHUA. “Learning deep architectures for AI.” **Foundations and trends® in Machine Learning** 2.1 (2009): 1-127.

BIRD, S., KLEIN, E., LOPER, E. **Natural language processing with Python: analyzing text with the natural language toolkit**. O'Reilly Media, Inc.", 2009.

BMF. Bolsa de Valores de São Paulo e da Bolsa de Mercadorias e Futuros, 2016. Disponível em: http://www.bmfbovespa.com.br/pt_br/institucional/sobre-a-bm-fbovespa/visite-a-bm-fbovespa/. Acesso em: 20 jul 2016.

BMF2. Bolsa de Valores de São Paulo e da Bolsa de Mercadorias e Futuros, 2016. Disponível em: http://ri.bmfbovespa.com.br/static/ptb/empresas-do-grupo.asp?idioma=ptb. Acesso em: 20 jul 2016.

BMF3. Bolsa de Valores de São Paulo e da Bolsa de Mercadorias e Futuros. Disponível em: http://bvmf.bmfbovespa.com.br/download/BOLETINSDIARIOS/boletimdiario_20170727.pdf. Acesso em: 20 jul 2016.

BOULIS, C., OSTENDORF, M., “Text Classification by Augmenting the Bag-of-Words Representation with Redundancy Compensated Bi-grams”. In: **Proceedings of the International Workshop in Feature Selection in Data Mining**, pp. 9-16, 2005.

BOUREAU, Y, AND YANN L. CUN. “Sparse feature learning for deep belief networks.” **Advances in neural information processing systems**. 2008.

BREIDT, E. “Extraction of VN-collocations from text corpora: A feasibility study for German”. **arXiv preprint cmp-lg/9603006**, 1996.

BULLINARIA, J., LEVY, J. “Extracting semantic representations from word cooccurrence statistics: A computational study”. **Behavior Research Methods**, 39 (3), 510–526 (2007).

CAMACHO-COLLADOS, J., PILEHVAR, M. T. “On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis”. **arXiv preprint arXiv:1707.01780**, 2017.

Cao, W., Gao, J., Ming, Z., & Cai, S. “Some Tricks in Parameter Selection for Extreme Learning Machine”. **In IOP Conference Series: Materials Science and Engineering** (Vol. 261, No. 1, p. 012002). IOP Publishing (2017, October)

CHURCH, K. W. E HANKS, P. “Word association norms, mutual information and lexicography”. **Computational linguistics**, 16(1), 22-29, 1989.

COLBY, K. M. “Modeling a paranoid mind.” **Behavioral and Brain Sciences** v.4, n.4: 515-534, 1981.

COLLOBERT, R., WESTON J. “A unified architecture for natural language processing: Deep neural networks with multitask learning.” **Proceedings of the 25th international conference on Machine learning**. ACM, 2008.

COLLOBERT, R., WESTON, J., BOTTOU, L., KARLEN, M., KAVUKCUOGLU, K., KUKSA, P. “Natural language processing (almost) from scratch”. **Journal of Machine Learning Research**, 12(Aug), 2493-2537, 2011.

COOK, T. R., SMALTER H., A. **Macroeconomic Indicator Forecasting with Deep Neural Networks**. (No. RWP 17-11), 2017.

CORTES, C., AND VAPNIK V. “Support-vector networks.” **Machine learning** v.20, v.3 (1995): 273-297.

DEERWESTER, S., DUMAIS, S. T., FURNAS, G. W., LANDAUER, T. K., HARSHMAN, R. “Indexing by latent semantic analysis.” **Journal of the American society for information science** 41.6 (1990): 391.

DENG, W., QINGHUA Z., AND LIN C. “Regularized extreme learning machine.” **Computational Intelligence and Data Mining**, 2009. CIDM'09. IEEE Symposium on. IEEE, 2009.

DING, X., ZHANG, Y., LIU, T., DUAN, J. “Deep Learning for Event-Driven Stock prediction”. **Proceedings of the 24th International Joint Conference on Artificial Intelligence** (ICJAI’15). 2015.

DING, S., ZHAO, H., ZHANG, Y., XU, X., NIE, R. “Extreme learning machine: algorithm, theory and applications.” **Artificial Intelligence Review** 44.1: 103-115, 2015b.

DOS SANTOS P., L., DRAS, M. “Stock Market Prediction with Deep Learning: A Character-based Neural Language Model for Event-based Trading.” **In Proceedings of the Australasian Language Technology Association Workshop** (pp. 6-15), 2017.

DUNNING, T. “Accurate methods for the statistics of surprise and coincidence”. **Computational linguistics**, 19(1), 61-74, 1993.

EVANGELOPOULOS, N., ZHANG, X., PRYBUTOK, V. R. “Latent semantic analysis: five methodological recommendations”. **European Journal of Information Systems**, 21(1), 70-86, 2012.

EVERT, S. “The statistics of word cooccurrences: word pairs and collocations”, 2005.

FAMA, E., F., **The Distribution of the Daily Differences of the Logarithms of Stock Prices**. Ph.D. Universidade de Chicago. Chicago, 1964.

FAN, Z., AND LING SHAO. “Weakly-supervised cross-domain dictionary learning for visual recognition.” **International Journal of Computer Vision** 109.1-2 (2014): 42-59.

FARIA, E., L., ALBUQUERQUE, M., P, GONZALEZ, J. L., et al. “Predicting the Brazilian stock market through neural networks and adaptive exponential smoothing methods”. **Expert System with Applications** v. 36, pp. 12506-12509, May. 2009.

FARIA, E. L. **Uma metodologia para a previsão do índice Bovespa utilizando mineração de textos**. Dissertação de mestrado M.Sc., Universidade Federal do Rio de Janeiro. Rio de Janeiro, Julho 2012.

FEHRER, R., FEUERRIEGEL, S. "Improving Decision Analytics with Deep Learning: The case of financial disclosures". **Arxiv Preprint Arxiv: 1508.01993** (2015)

FENG, S., WANG, Y., SONG, K., WANG, D., YU, G. "Detecting Multiple Coexisting Emotions in Microblogs with Convolutional Neural Networks". **Cognitive Computation**, 10(1), 136-155, 2018.

FREUND, YOAV, DAVID HAUSSLER. "Unsupervised learning of distributions on binary vectors using two layer networks." **Advances in neural information processing systems**. 1992.

FUKUSHIMA, K. "Cognitron: A self-organizing multilayered neural network." **Biological cybernetics** 20.3-4 (1975): 121-136.

FUKUSHIMA, K. "Training multi-layered neural network Neocognitron". **Neural Networks**, V. 40 (2013), 18–31.

GIDOFALVI, G., ELKAN, C. "Using news articles to predict stock price movements". **Department of Computer Science and Engineering, University of California, San Diego**, 2001.

GRAVES, A., MOHAMED, A., HINTON, G., E. "Speech recognition with deep recurrent neural networks". **IEEE international conference on acoustics, speech and signal processing**. IEEE, 2013.

GLOROT, X., BENGIO, Y. "Understanding the difficulty of training deep feedforward neural networks". **In International conference on artificial intelligence and statistics** (pp. 249-256), 2010.

GLOROT, X., BORDES, A., BENGIO, Y. “Deep sparse rectifier neural networks”. **In International Conference on Artificial Intelligence and Statistics** (pp. 315-323), 2011.

GOADRICH, M., OLIPHANT, L., SHAVLIK, J. “Gleaner: Creating Ensembles of First-Order Clauses to Improve Recall-Precision Curves”, **Journal of Machine Learning**, v. 64, Issue: 1-3, pp. 231-261, Sep. 2006.

GOLUB G H, VAN LOAN C F. **Matrix Computation**. 3rd Ed. Baltimore: Johns Hopkins Univ. Press; 1996.

GOODFELLOW I., BENGIO, Y., Courville, A. **Deep Learning** (Vol.1). Cambridge: MIT Press (2016).

GUTMANN, M. U., HYVÄRINEN, A. “Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics”. **Journal of Machine Learning Research**, 13(Feb), 307-361, (2012).

HAFEMANN, LUIZ G., OLIVEIRA, L. S., CAVALIN, P. R., SABOURIN, R. “Transfer learning between texture classification tasks using Convolutional Neural Networks.” **International Joint Conference on Neural Networks (IJCNN)**. IEEE, 2015.

HARRIS, Z. S. “Distributional structure”. **Word** 10.2-3 (1954): 146-162

HAYKIN, S. **Redes Neurais, Princípios e Prática**. 2.ed. Porto Alegre: Bookman, 2001.

HEBB, D. O. **The organization of behavior: A neuropsychological theory**. New York: Wiley, 1949.

HINTON, G. E. “Learning distributed representations of concepts.” **Proceedings of the eighth annual conference of the cognitive science society**. Vol. 1. 1986.

HINTON, G. E., OSINDERO, S., THE, Y. W, “A fast learning algorithm for deep belief nets,” **Neural Computation**, vol. 18, pp. 1527–1554, 2006.

HINTON, G. E., SALAKHUTDINOV, R., R. “Reducing the dimensionality of data with neural networks”. **Science** 313.5768 (2006), 504–507.

HINTON, G. E. “Training products of experts by minimizing contrastive divergence.” **Neural computation**, 14(8), 1771-1800 (2006).

HINTON, G. E., RUSLAN R. S. “Using deep belief nets to learn covariance kernels for Gaussian processes.” **Advances in neural information processing systems**. 2008.

HINTON, G. E., SRIVASTAVA, N., KRIZHEVSKY, A., SUTSKEVER, I., SALAKHUTDINOV, R. R. “Improving neural networks by preventing co-adaptation of feature detectors”. **arXiv preprint arXiv:1207.0580** (2012).

HOCHREITER, S., SCHMIDHUBER, J. “Long short-term memory.” **Neural computation** 9.8 (1997): 1735-1780.

HOERL, A. E., ROBERT, W. K. “Ridge regression: biased estimation for nonorthogonal problems.” **Technometrics** 42.1 (2000): 80-86.

HOPFIELD, J. J. “Neural networks and physical systems with emergent collective computational abilities.” **Proceedings of the national academy of sciences** 79.8 (1982): 2554-2558.

HOPFIELD, J. J. “Neurons with graded response have collective computational properties like those of two-state neurons.” **Proceedings of the national academy of sciences** 81.10 (1984): 3088-3092.

HUANG, G. B., BABRI, H. A. “Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions.” **IEEE Transactions on Neural Networks** 9.1 (1998): 224-229.

HUANG, G. B., ZHU, Q. Y., SIEW, C.-K. “Extreme learning machine: a new learning scheme of feedforward neural networks.” **Neural Networks. Proceedings. 2004 IEEE International Joint Conference on**. Vol. 2. IEEE, 2004.

HUANG, G. B., ZHU, Q. Y., SIEW, C.-K. “Extreme learning machine: Theory and applications”. **Neurocomputing**, 70.1 (2006): 489 - 501.

HUANG, G. B, WANG, D. H., E LAN., Y. “Extreme learning machines: a survey.” **International journal of machine learning and cybernetics** 2.2 (2011): 107-122.

HUANG, G. B, ZHOU, H., DING, X., ZHANG, R. “Extreme learning machine for regression and multiclass classification.” **IEEE Transactions on Systems, Man, and Cybernetics**, Part B (Cybernetics) 42.2 (2012): 513-529.

HUBEL, DAVID H., TORSTEN N. WIESEL. “Receptive fields, binocular interaction and functional architecture in the cat's visual cortex.” **The Journal of physiology** 160.1 (1962): 106-154.

HUBEL, DAVID H., TORSTEN N. WIESEL. “Ferrier lecture: Functional architecture of macaque monkey visual cortex.” **Proceedings of the Royal Society of London B: Biological Sciences** 198.1130 (1977): 1-59.

HUGHES, M., LI, I., KOTOULAS, S., SUZUMURA, T. “Medical text classification using convolutional neural networks”. **Stud Health Technol Inform**, 235, 246-50, (2017).

HUYNH, H. D., DANG, L. M., DUONG, D. “A New Model for Stock Price Movements Prediction Using Deep Neural Network”. **In Proceedings of the Eighth International Symposium on Information and Communication Technology** (pp. 57-62). ACM, 2017

IVAKHNENKO, A. G. “The group method of data handling-A rival of the method of stochastic approximation”. **Soviet Automatic Control**, Vol.1, N. 3, 1968, 43–55.

IVAKHNENKO, A. G. “Polynomial theory of complex systems.” **IEEE transactions on Systems, Man, and Cybernetics** 1.4 (1971): 364-378.

IVAKHNENKO, A. G., LAPA, V. G. **Cybernetic predicting devices**. No. TR-EE66-5. PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGINEERING, 1966.

IVAKHNENKO, A. G., LAPA, V. G., MCDONOUGH, R. N. “Cybernetics and forecasting techniques”. **Elsevier**, New York (1967).

JARRETT, K., KAVUKCUOGLU, K., RANZATO, M. A., LECUN, Y. What is the best multi-stage architecture for object recognition? In **Computer Vision, 2009 IEEE 12th International Conference on** (pp. 2146-2153). IEEE (2009)

JURAFSKY, D., JAMES H. MARTIN. **Speech and language processing**. Vol. 3. London: Pearson, 2014.

KALCHBRENNER, N., GREFENSTETTE, E., BLUNSOM, P. “A convolutional neural network for modelling sentences”. **arXiv preprint arXiv**:1404.2188, 2014.

KEOGH, E., CHU, S., HART, D., PAZZANI, M. “An online algorithm for segmenting time series”. **In Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on** (pp. 289-296), 2001.

KIM, Y. “Convolutional neural networks for sentence classification”. **arXiv preprint arXiv:1408.5882**, 2014.

KINGMA, D. P., BA, J. “Adam: A method for stochastic optimization”. **In Proceedings of the 3rd International Conference on Learning Representations** (ICLR), 2014.

KORCZAK, J., HEMES, M. “Deep learning for financial time series forecasting in A-Trader system”. **In Computer Science and Information Systems (FedCSIS)**, **2017 Federated Conference on** (pp. 905-912). IEEE, 2017.

KRIZHEVSKY, A., SUTSKEVER, I., HINTON, G., E. “ImageNet Classification with Deep Convolutional Neural Netwoks”. **Advances in neural information processing systems**. 2012.

KWOK, T. Y., YEUNG, D. Y. “Objective functions for training new hidden units in constructive neural networks.” **IEEE Transactions on neural networks 8.5** (1997): 1131-1148.

LAN, Y., SOH, Y. C., HUANG, G. B. "Ensemble of online sequential extreme learning machine." **Neurocomputing** 72.13-15 (2009): 3391-3395.

LAROCHELLE, H., ERHAN, D., COURVILLE, A., BERGSTRA, J., BENGIO, Y. "An empirical evaluation of deep architectures on problems with many factors of variation." **Proceedings of the 24th international conference on Machine learning**. ACM, 2007.

LAVRENKO, V., SCHMILL, M., LAWRIE, D., OGILVIE, P., JENSEN, D., ALLAN, J. "Language models for financial news recommendation". In **Proceedings of the ninth international conference on Information and knowledge management** (pp. 389-396). ACM, 2000.

LECUN, Y., BOSER, B., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W., JACKEL, L. D. "Backpropagation applied to handwritten zip code recognition." **Neural computation** 1.4 (1989): 541-551.

LECUN, Y., BOTTOU, L., BENGIO, Y., HAFFNER, P. "Gradient-based learning applied to document recognition." **Proceedings of the IEEE** 86.11 (1998): 2278-2324.

LEE, L. W., CHEN, S. M. "New methods for text categorization based on a new feature selection method and a new similarity measure between documents". **In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems** (pp. 1280-1289). Springer, Berlin, Heidelberg, 2006.

LIANG, N. Y., HUANG, G. B., SARATCHANDRAN, P., SUNDARARAJAN, N. "A fast and accurate online sequential learning algorithm for feedforward networks." **IEEE Transactions on neural networks** 17.6 (2006): 1411-1423.

LIU, H. S., B. Y. LEE, AND Y. S. TARNG. "In-process prediction of corner wear in drilling operations." **Journal of Materials Processing Technology** 101.1 (2000): 152-158.

LYSE, G. I., ANDERSEN, G. "Collocations and statistical analysis of n-grams". **Exploring Newspaper Language: Using the Web to Create and Investigate a Large Corpus of**

Modern Norwegian, Studies in Corpus Linguistics, John Benjamins Publishing, Amsterdam, 79-109, 2012.

MARTÍNEZ-MARTÍNEZ, J. M., ESCANDELL-MONTERO, P., SORIA OLIVAS, E., MARTÍN GUERRERO, J. D., MAGDALENA-BENEDITO, R., GÓMEZ-SANCHIS, J. "Regularized extreme learning machine for regression problems." **Neurocomputing** 74.17 (2011): 3716-3721.

MAURO, H. **Investimentos: Como administrar melhor seu dinheiro.** 1 ed. São Paulo, Editora Fundamento Educacional, 2001.

MCCULLOCH, WARREN S., WALTER PITTS. "A logical calculus of the ideas immanent in nervous activity." **The bulletin of mathematical biophysics** 5.4 (1943): 115-133.

MIKOLOV, T., CHEN, K., CORRADO, G., DEAN, J. "Efficient estimation of word representations in vector space". **arXiv preprint arXiv:1301.3781** (2013a).

MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. S., DEAN, J. "Distributed representations of words and phrases and their compositionality". **In Advances in neural information processing systems**, pgs.3111–3119, (2013b).

MINSKY, M. L. Papert, S. A., **Perceptrons**, Cambridge, MA: MIT Press.

MNIH, A., TEH, Y. W. "A fast and simple algorithm for training neural probabilistic language models". **arXiv preprint arXiv:1206.6426**, 2012.

MORIN, FREDERIC, YOSHUA BENGIO. "Hierarchical Probabilistic Neural Network Language Model." **Aistats**. Vol. 5, pp. 246-252, 2005.

NAIR, V., HINTON, G. E. "Rectified linear units improve restricted boltzmann machines". **In Proceedings of the 27th International Conference on Machine Learning (ICML-10)** (pp. 807-814), 2010.

NAVON, A., KELLER, Y. “Financial Time Series Prediction Using Deep Learning”. **arXiv preprint arXiv:1711.04174**, 2017.

NIELSEN, M. “Deep learning.” **In: Neural Networks and Deep Learning**. Michael Nielsen, 2016. cap. 6. Disponível em: http://neuralnetworksanddeeplearning.com/chap6.html

PARKER, JIM R. **Algorithms for image processing and computer vision**. John Wiley & Sons, 2010.

PEARCE, D. “A Comparative Evaluation of Collocation Extraction Techniques”. In **LREC**, 2002.

PEDERSEN, T. “Fishing for exactness”. **arXiv preprint cmp-lg/9608010**, 1996.

PENG, Y., JIANG, H. “Leverage financial news to predict stock price movements using word embeddings and deep neural networks”. **arXiv preprint arXiv:1506.07220**, 2015.

PENNINGTON, J., SOCHER, R., MANNING, C. “Glove: Global vectors for word representation”. In **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)** (pp. 1532-1543), 2014.

RADIM, R., SOJKA, P. “Software Framework for Topic Modelling with Large Corpora”. **Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks**, p.45-50, 2010.

RADINSKY, K., DAVIDOVICH, S., MARKOVITCH, S. “Learning causality for news events prediction”. In **Proceedings of the 21st international conference on World Wide Web (pp. 909-918)**. ACM, 2012.

RAO, C. R.; MITRA, S. K. “Generalized inverse of a matrix and its applications”. **Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics**, 601-620, University of California Press, Berkeley, Calif., 1972. https://projecteuclid.org/euclid.bsmsp/1200514113

REYNOLDS, A. CRAIG. “The conference on mechanical translation.” **Mechanical Translation** v.1, n.3, p. 47–55 (1954).

REZENDE, S. O. **Sistemas inteligentes: fundamentos e aplicações**. Editora Manole Ltda, 2003.

RONG, X. “Word2vec parameter learning explained”. **arXiv preprint arXiv:1411.2738**, 2014.

ROSENBLATT, FRANK. “The perceptron: A probabilistic model for information storage and organization in the brain.” **Psychological review** 65.6 (1958): 386.

RIE, J., AND ZHANG, T. “Effective use of word order for text categorization with convolutional neural networks.” **arXiv preprint arXiv:1412.1058** (2014).

RUIZ, E. J., HRISTIDIS,V., CASTILLO, C., GIONIS, A., JAIMES, A. “Correlating financial time series with micro-blogging activity.” **Proceedings of the fifth ACM international conference on Web search and data mining. ACM**, 2012.

RUMELHART, D. E., HINTON, G. E., WILLIAMS, R.J. “Learning representations by back-propagating errors.” **Nature** 323.6088 (1986): 533-538.

RUPERT, J. F. **Bulletin of the School of Oriental and African Studies**. Cambridge University Press. V.24, pages:412-418 (1961).

SAG, I. A., BALDWIN, T., BOND, F., COPESTAKE, A., FLICKINGER, D. “Multiword expressions: A pain in the neck for NLP”. In **International Conference on Intelligent Text Processing and Computational Linguistics (pp. 1-15)**. Springer, Berlin, Heidelberg, 2002.

SALTON, G., WONG A., YANG C. S. “A vector space model for automatic indexing.” **Communications of the ACM** 18.11 (1975): 613-620.

SCHNABEL, T.; LABUTOV, I.; MIMNO, D.; THORSTEN, J. “Evaluation methods for unsupervised word embeddings”. In: **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing**. 2015. p. 298-307, 2015.

SCHONE, P., JURAFSKY, D. “Is knowledge-free induction of multiword unit dictionary headwords a solved problem?”. In **Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing**, 2001.

SCHUMAKER, R., P. CHEN, H. “A quantitative stock prediction system based on financial news”. **Information Processing and Management** v.45, pp. 571-583, Apr. 2009.

SERRE, D. **Matrices: Theory and Applications**. Springer, 1st edition. 2002

SHEN, Y., HE, X., GAO, J., DENG, L., MESNIL, G. “Learning semantic representations using convolutional neural networks for web search”. In **Proceedings of the 23rd International Conference on World Wide Web** (pp. 373-374). ACM, 2014.

STORN, R., PRICE, K. “Differential evolution a simple and efficient heuristic for global optimization over continuous spaces.” **Journal of global optimization** 11.4 (1997): 341-359.

TIBSHIRANI, R. “Regression shrinkage and selection via the lasso: a retrospective”. **Journal of the Royal Statistical Society: Series B (Statistical Methodology)** 73.3 (2011): 273-282.

TURNEY, P. D., PANTEL P. “From frequency to meaning: Vector space models of semantics.” **Journal of artificial intelligence research** 37 (2010): 141-188.

TURING, A. “Computing machinery and intelligence.” **Mind**, v.59, n.236, p.433–460 (1950).

VARGAS, M. R., DE LIMA, B. S., EVSUKOFF, A. G. “Deep learning for stock market prediction from financial news articles”. In **Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)**, 2017 IEEE International Conference on (pp. 60-65). IEEE.

YALCINKAYA, M., SINGH, V. “Patterns and trends in building information modeling (BIM) research: a latent semantic analysis”. **Automation in Construction**, 59, 68-80, 2015.

YIH, W. T., HE, X., MEEK, C. “Semantic parsing for single-relation question answering”. In **Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics** (Volume 2: Short Papers) (Vol. 2, pp. 643-648), 2014.

YOSHIHARA, A., SEKI, K., UEHARA, K. “Leveraging temporal properties of news events for stock market prediction”. **Artificial Intelligence Research**, 5(1), 103, 2015.

WANG, B., HUANG, H., WANG, X. “A novel text mining approach to financial time series forecasting.” **Neurocomputing** 83: 136-145, 2012.

WANG, Y., WANG, X. J. “A new approach to feature selection in text classification”. In Machine Learning and Cybernetics. **Proceedings of 2005 International Conference on** (Vol. 6, pp. 3814-3819). IEEE, 2005.

WEISS, D., ALBERTI, C., COLLINS, M., PETROV, S. “Structured training for neural network transition-based parsing”. **arXiv preprint arXiv:1506.06158**, 2015.

WEIZENBAUM, J. **Computer power and human reason: From judgment to calculation**. New York: W.H. Freman and Company. Pp. 2,3,6,182,189. (1976). ISBN 0-7167-0464-1

WINOGRAD, T. **Procedures as a representation for data in a computer program for understanding natural language**. No. MAC-TR-84. MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1971.

VAPINICK, V., N. **Statistical Learning Theory**. New York (NY): Wiley-Inter Science, 1998.

ZHANG, Y., WALLACE, B. “A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification”. **arXiv preprint arXiv:1510.03820** (2015).

ZHANG, X., ZHAO, J., LECUN, Y. “Character-level Convolutional Networks for Text classification”. **Advances in Neural Information Processing Systems**. 2015

ZHANG, X., ZHAO, J., LECUN, Y. “Text Understanding from Scratch”. **arXiv preprint arXiv:1502.01710** (2015).

ZHU, Q. Y., QIN, A. K., SUGANTHAN, P. N., HUANG, G. B. “Evolutionary extreme learning machine.” **Pattern recognition** 38.10 (2005): 1759-1763.

ZOU, H., TREVOR, H. “Regularization and variable selection via the elastic net.” **Journal of the Royal Statistical Society: Series B (Statistical Methodology)** 67.2 (2005): 301-320.