Marta Talitha Carvalho Freire Mendes

Detecção de Novidades em Mídias Sociais pela Fusão de Texto e Imagem em uma Única Estrutura de Dados

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico - CT

Programa de Pós-Graduação em Engenharia Elétrica – PPGEE

Orientador: Patrick Marques Ciarelli

Vitória

2020

Marta Talitha Carvalho Freire Mendes
Detecção de Novidades em Mídias Sociais pela Fusão de Texto e Imagem em uma Única Estrutura de Dados/ Marta Talitha Carvalho Freire Mendes. – Vitória, 2020-
?? p. : il. (algumas color.) ; 30 cm.

Orientador: Patrick Marques Ciarelli

Tese de Doutorado – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico - CT
Programa de Pós-Graduação em Engenharia Elétrica – PPGEE , 2020.
1.

Marta Talitha Carvalho Freire Mendes

Detecção de Novidades em Mídias Sociais pela Fusão de Texto e Imagem em uma Única Estrutura de Dados

Trabalho aprovado. Vitória, 17 de agosto de 2020:

Patrick Marques Ciarelli
Orientador
Universidade Federal do Espírito Santo

Adrião Duarte Doria Neto
Membro Externo
Universidade Federal do Rio Grande do Norte

Claudine Badue
Membro Externo
Universidade Federal do Espírito Santo

Elias Silva de Oliveira
Membro Externo
Universidade Federal do Espírito Santo

Luiz Alberto Pinto
Membro Externo
Instituto Federal do Espírito Santo

Vitória
2020

"Dedico este trabalho a El Shaddai , meu criador, meu sustentador e minha força."

Agradecimentos

Agradeço a Deus, por estar comigo durante essa caminhada, me iluminando e concedendo graça em momentos difíceis da pesquisa. Certamente, Deus me propiciou concluir este trabalho com sucesso.

Ao meu amado esposo Warley Rocha Mendes e minha amada filha Sarah Freire Mendes pelo amor, compreensão e por toda ajuda ao longo deste trabalho.

Aos meus pais Gerson Freire de Amorim Filho e Rosangela Carvalho pelo incentivo, pelas orações e carinho. A minha irmã e amiga Isabela Carvalho Freire de Amorim pelo incentivo e orações. A minha irmã e amiga Evelin Carvalho Freire de Amorim que muito me apoiou com seus conselhos e trocas de ideias.

Ao meu orientador Patrick Marques Ciarelli por suas dicas e orientações. Aos professores membros da Banca Examinadora, por terem atendido ao convite para desempenhar este papel, dispondo de seu tempo e conhecimento para analisar este trabalho.

Ao professor da engenharia de produção da UFES, Frederico Damasceno Bortoloti por sua amizade, parceria e toda ajuda ao longo do trabalho. Ao professor Daniel Cavalieri do IFES, pelas ideias e sugestões para a construção desta tese. Aos professores do curso de estatística da UFES, Alexandres Loureiro Rodrigues e Mauro Cesár Martins Campos pelo tempo disponibilizado para esclarecimento de dúvidas importantes para construção deste trabalho.

Aos meus amigos do PPGEE, Vinicius Ávila, Gentil Auer Neto e Marisa Matedi pelo apoio durante a realização deste curso.

Resumo

As mídias sociais têm desempenhado um papel muito importante na detecção de novidades ou eventos, pois se mostram vastamente disponíveis, além de possibilitarem rápida propagação de diversos tipos de informações. Contudo, os dados são não estruturados, então surge o desafio de minerar eventos em uma grande massa de dados que está constantemente em crescimento. Diante disso, é necessário não apenas identificar informações, mas também detectar as informações mais relevantes. A relevância de uma informação pode estar relacionada com vários aspectos e características das aplicações, tais como: temporal, audiência, contexto, entre outras. Tais aplicações procuram identificar padrões novos ou não familiares em conjuntos de dados, alguns exemplos são: cobertura de notícias, tendência de produtos, comportamentos suspeitos em mídias sociais para detecção de crimes, entre outras.

Esta pesquisa propõe duas novas arquiteturas de detecção de novidades em dados de redes sociais, baseadas em três pilares: fusão de dados, janelas temporais e um modelo para qualificar a audiência da novidade. Ambas as arquiteturas possuem em comum os seguintes *pipelines*: codificação, fusão e detecção. A codificação utiliza redes neurais para representar os dados não estruturados em vetores densos. A fusão transforma distintas estruturas de dados em uma única estrutura ou faz a combinação das pontuações (ou probabilidades) na saída dos classificadores. A detecção utiliza algoritmos não supervisionados para identificar as novidades.

A principal diferença entre as duas arquiteturas está na fusão. A primeira arquitetura realiza a fusão dos dados na entrada dos algoritmos não supervisionados. A segunda arquitetura realiza a fusão na saída dos algoritmos não supervisionados. Os seguintes algoritmos não supervisionados são utilizados: HBOS, *Feature Bagging*, *Isolation Foresting*, *Autoencoders*, e versões não supervisionadas do kNN e do LSTM. A principal contribuição neste trabalho é uma abordagem baseada em redes neurais profundas que realiza fusão de dados e detecção de novidade em dados não estruturados e heterogêneos. As inovações do trabalho estão na criação de uma nova arquitetura para detecção de novidades das redes sociais, realizar uma nova fusão com dados não estruturados, criação de uma base de dados com dados não estruturados para a tarefa de detecção de novidades, e um método livre de contexto para definir o que é novidade. A etapa de experimentos mostrou que a fusão proposta teve uma melhoria de aproximadamente 11% em relação ao não uso da fusão na detecção de novidades.

Palavras-chave: Detecção de Novidade, Métodos de Fusão, Aprendizado Profundo, Rede Social, Fluxo Contínuo de Dados.

Abstract

Social media has played a very important role in detecting novelties or events, as they are widely available, in addition to allowing the rapid spread of various types of information. However, the data are unstructured, so the challenge arises of mining events in a large mass of data that are constantly growing. Therefore, it is necessary not only to identify information, but also to detect the most relevant information. The relevance of an information can be related to several aspects and characteristics of the applications, such as: time, audience, context, among others. Such applications seek to identify new or unfamiliar patterns in data sets, some examples are: news coverage, product trends, suspicious behaviors on social media to detect crimes, among others.

This research proposes two new novelty detection architectures in social media data based on three pillars: data fusion, temporal windows and a model to qualify the audience for the novelty. Both architectures have the following pipelines in common: coding, merging and detection. Coding uses neural networks to represent unstructured data in dense vectors. The fusion transforms different data structures into a single structure or combines the scores (or probabilities) in the classifiers output. Detection uses unsupervised algorithms to identify novelties.

The main difference between the two architecture is in the fusion. The first architecture fuses data at the input of unsupervised algorithms. The second architecture fuses the output of unsupervised algorithms. The following unsupervised algorithms are used: HBOS, Feature Bagging, Isolation Foresting, Autoencoders, and unsupervised versions of kNN and LSTM. The main contribution in this work is an approach based on deep neural networks that performs data fusion and novelty detection in unstructured and heterogeneous data. The innovations in the work are in the creation of a new architecture for detecting news from social networks, making a new merger with unstructured data, creating a database with unstructured data for the task of detecting news, and a context-free method for defining what is a novelty. The experiment stage showed that the proposed fusion had an improvement of approximately 11% in relation to the non-use of the fusion in detecting novelties.

Keywords: Novelty Detection, Fusion Methods, Deep Learning, Social Media, Data Stream.

Lista de ilustrações

Figura 1 – A figura a) representa um imagem normal e b) representa uma novidade. 26
Figura 2 – Matriz de ocorrências. 27
Figura 3 – Grafo de correlação entre os tipos de dados. 29
Figura 4 – Modelo de rede neural para dados heterogêneos geral. 31
Figura 5 – Encadeamento de eventos sobre o tempo. 33
Figura 6 – O modelo proposto por Alkhodair et al. (2019) para detectar rumor. Um tweet w é primeiro tokenizado em uma sequência de palavras $w =< w_1, \ldots, w_T >$. Próximo passo, o modelo word2vec converte a sequência de palavras em sequência de vetores $x =< x_1, \ldots, x_T >$, que são entradas da rede LSTM-RNN. Finalmente, a rede LSTM-RNN prediz a classe de saída no tempo T. 34
Figura 7 – Explicação gráfica da decomposição fundamental do erro. 40
Figura 8 – Erro do tipo viés. 40
Figura 9 – Erro do tipo variância. 41
Figura 10 – Ilustração gráfica do viés e variância. 42
Figura 11 – Tipos de *ensembles*. 43
Figura 12 – Exemplo da operação de convolução 2-D. 45
Figura 13 – Camadas convolucionais. 46
Figura 14 – Arquitetura de uma rede neural convolucional (*Faster* R-CNN). 48
Figura 15 – A figura da esquerda mostra o resultado da técnica de detecção de objetos e a figura da direita mostra o resultado da técnica de segmentação de imagens. 49
Figura 16 – A figura da esquerda mostra a técnica de segmentação semântica e a figura da direita mostra a técnica de segmentação de instâncias. 49
Figura 17 – Máscaras coloridas, *bounding boxes* e as probabilidades de confiança geradas pela *Mask R-CNN*. 50
Figura 18 – Um exemplo de detecção de sinal de trânsito. O *bounding box* predito é traçado em vermelho enquanto o *bounding box ground-truth* é traçado em verde. O objetivo é calcular a área de interseção sobre a união destes *bounding boxes*. 51
Figura 19 – O cálculo da interseção sobre a união é uma simples divisão das áreas dos *bounding boxes*. 51
Figura 20 – Imagem com o *bounding box ground-truth*. 52
Figura 21 – Imagem com os *bounding boxes* candidatos ou regiões preditas. 52
Figura 22 – Exemplo de aprendizado supervisionado. 54

Figura 23 – Um conjunto de dados envolvendo três *clusters*. Cada *cluster* é mostrado usando um símbolo colorido diferente. Esquerda: Os três *clusters* estão bem separados. Nesse cenário, uma abordagem de *cluster* deve identificar com êxito os três grupos. Direita: Há uma sobreposição entre os grupos. A tarefa de identificar os *clusters* é mais desafiadora. 55
Figura 24 – Pequenos grupos de anomalias que podem ser um desafio para a correta identificação. 57
Figura 25 – Diferentes projeções em duas dimensões do espaço podem ou não revelar diferentes conjuntos de *outliers*. 60
Figura 26 – Esboço do *autoencoder* totalmente conectado usado na proposta deste trabalho. 66
Figura 27 – Esboço da arquitetura da *LSTM*. 67
Figura 28 – Arquitetura de um modelo de linguagem neural. 70
Figura 29 – Exemplo da combinação linear da entrada. 71
Figura 30 – Exemplo da combinação linear da camada oculta. 72
Figura 31 – Probabilidade das palavras do Corpus. 72
Figura 32 – Vetores densos de países e capitais projetados usando a técnica PCA. . 73
Figura 33 – Exemplo de um grafo. 74
Figura 34 – Exemplo de um *tweet* e seus metadados. 78
Figura 35 – *Tweets* referentes ao intervalo 21/04/2018 a 01/05/2018. As setas destacam os eventos novidades, e que são representados pelos vértices em azul. As não novidades são representadas pelos vértices em vermelho. . 82
Figura 36 – *Tweets* referentes ao intervalo 01/04/2018 a 11/04/2018. As setas destacam os eventos novidades, e que são representados pelos vértices em azul. As não novidades são representadas pelos vértices em vermelho. . 83
Figura 37 – Rede de codificação do texto (*Encoder*). 84
Figura 38 – Distribuição do tamanho dos *tweets*. 85
Figura 39 – Transformação de um *tweet* para vetor denso, i.e., representação em *word embedding*. 87
Figura 40 – Rede de codificação da imagem (*Encoder*). 88

Figura 41 – O *pipeline* das duas arquiteturas propostas para detecção de novidades em tempo real. A arquitetura na parte (a) funde o texto e a imagem usando a nova abordagem proposta por este trabalho usando o *Dataset* COCO e a rede convolucional *MASK R-CNN*. Primeiro, a fusão dos dados é realizada a partir dos rótulos gerados pela rede *MASK R-CNN*. Após, o resultado da fusão é codificado e finalmente a novidade é classificada usando algoritmos não supervisionados. A arquitetura na parte (b) realiza a codificação separadamente, classifica com algum algoritmo não supervisionado e finalmente realiza a fusão dos dados com o algoritmo AOM. 90
Figura 42 – Exemplo de uma imagem de um *tweet* em que a rede pré-treinada *MASK R-CNN* reconhece os seguintes objetos em cena: *person* (0,999), *tie* (0,998), *clock* (0,952), *book* (0,346) e um *laptop* (0,917). 91
Figura 43 – Exemplo de uma imagem de um *tweet* em que a rede pré-treinada *MASK R-CNN* reconhece os seguintes objetos em cena: *person* (0,996), *person* (0,999) e *person* (0,994). 91
Figura 44 – Exemplo de uma imagem de um *tweet* em que a rede pré-treinada *MASK R-CNN* reconhece os seguintes objetos em cena: *train* (0,564), *car* (0,994), *person* (0,975), *fire hydrant* (0,750), *car* (0,963), *person* (0,994), *car* (0,716), *truck* (0,875) e *train* (0,564). 92
Figura 45 – Exemplo de uma imagem de um *tweet* em que a rede pré-treinada *MASK R-CNN* reconhece os seguintes objetos em cena: *boat* (0,632), *boat* (0,979), *boat* (0,995), *boat* (0,521), *chair* (0,0687), *boat* (0,973) e *boat* (0,986). 92
Figura 46 – Curva ROC relacionando a taxa de verdadeiros positivos (TVP) contra a taxa de falsos positivos (TFP), em que o eixo x é TFP e o eixo y é TVP 94
Figura 47 – Esquema do treinamento, validação e teste na janela temporal. Na fase de treinamento e validação é possível obter o melhor parâmetro por meio da curva de aprendizado. 97
Figura 48 – Curva de aprendizado média para o algoritmo *IForest*. O melhor parâmetro *nestimators* médio para a nova técnica usando MASK R-CNN com janela 10 dias. 101
Figura 49 – Curva de aprendizado para o algoritmo kNN. O melhor parâmetro k médio para a nova técnica usando MASK R-CNN com janela de 10 dias. 101
Figura 50 – Curva de aprendizado para o algoritmo HBOS. O melhor parâmetro *nbins* médio para a nova técnica usando MASK R-CNN com janela de 10 dias. 102

Figura 51 – Curva de aprendizado para o algoritmo LSTM. O melhor parâmetro Época médio para a nova técnica usando MASK R-CNN com janela de 10 dias. 102
Figura 52 – Curva de aprendizado para o algoritmo *FBagging*. O melhor parâmetro *nestimators* médio para a nova técnica usando MASK R-CNN com a janela de 10 dias. 103
Figura 53 – Curva de aprendizado média para o algoritmo Autoencoder. O melhor parâmetro Época médio para a nova técnica usando MASK R-CNN com janela de 10 dias. 103
Figura 54 – Resultado para janelas temporais de 1, 5, 10 e 20 dias. 107
Figura 55 – Gráfico representando o erro de reconstrução de janelas temporais distintas. 108
Figura 56 – Exemplos de *tweets* com imagens nas quais usam a fusão com informações agregadas da base de dados COCO. Os seguintes objetos nas cenas são: (a) *dogs* (0,970); (0,994); (0,980) e *chair* (0,887); (b) *dog* (0,993); *birds* (0,987); (0,933); (0,917), *chair* (0,610) e *person* (0,500); (c) *person* (1,000), *tv* (0,968) e *tie* (0,540); (d) *person* (0,980), *tv* (0,989) e *tie* (0,803).111
Figura 57 – *Tweet* postado em 2018-06-12 01:08:35. 112

Lista de tabelas

Tabela 1 – Lista de métodos relatados na revisão da literatura 35
Tabela 2 – Algoritmos de aprendizado de máquina do estado da arte para detecção de novidades. 56
Tabela 3 – Exemplo de dicionário. 70
Tabela 4 – Algumas técnicas que geram vetores embutidos. 73
Tabela 5 – Matriz de confusão para a classificação de novidade ou não novidade . 94
Tabela 6 – Bases de dados da literatura. 95
Tabela 7 – Valores médios selecionados para os parâmetros dos algoritmos de aprendizado. 98
Tabela 8 – Resultados da métrica ROC AUC para janela temporal de 1 dia. . . . 104
Tabela 9 – Resultados da métrica ROC AUC para janela temporal de 5 dias. . . . 104
Tabela 10 – Resultados da métrica ROC AUC para janela temporal de 10 dias. . . 105
Tabela 11 – Resultados da métrica ROC AUC para janela temporal de 20 dias. . . 105
Tabela 12 – Resultados com diferentes proporções de texto e imagens usando a métrica ROC AUC para a janela temporal de 10 dias. 109
Tabela 13 – Tempo de treino e validação do algoritmo LSTM. 109
Tabela 14 – Uma amostra do conjunto de dados de *tweets* postados no dia 2018-06-12.112

Lista de abreviaturas e siglas

CIMM	*Consumption Intention Mining Model*
MCN	*Mean Cluster Novelty*
HFN	*High Novelty Fraction*
CNN	*Convolution Neural Network*
C-LSTM	*Contextual Long Short-Term Memory*
LSTM	*Long Short-Term Memory*
DNN	*Deep Neural Network*
NN	*Neural Network*
ReLU	*Rectified Linear Unit*
AOM	*Average of Maximum*
HBOS	*Histogram-Based Outlier Score*
kNN	*k Nearest Neighbors*
AUC	*Area Under the Curve*
RCNN	*Region-CNN*
ROC	*Receiver Operating Characteristic*
TP	*True Positive*
FP	*False Positive*
SURF	*Speeded Up Robust Features*
RGB	*Red-Green-Blue*
JSEG	*J value segmentation*
SIFT	*Scale-Invariant Feature Transform*
TF-IDF	*Term Frequency–Inverse Document Frequency*
HOG	*Histogram of Oriented Gradients*

GLCM	*Gray-Level Co-Occurrence Matrix*
NER	*Named Entity Recognizer*
DARPA	*Defense Advanced Research Projects Agency*
TDT	*Topic Detection and Tracking*
NED	*New Event Detection*
RED	*Retrospective Event Detection*
HMM	*Hidden Markov Model*
LDA	*Latent Dirichlet Allocation*

Lista de símbolos

X	Conjunto de dados de n instâncias
x	Exemplo (entrada) do conjunto de dados
y	Variável resposta (rótulo) associada com x para aprendizado supervisionado
n	Quantidade de dados em um conjunto de dados $n =\mid X \mid$
x_t	Exemplo cujas dimensões são características textuais
x_v	Exemplo cujas dimensões são características visuais
x_u	Exemplo cujas dimensões são características dos usuários
C	Evento composto pela união das características $< x_t, \cup x_v, \cup x_u >$
t_{ac}	Evento qualquer recém-chegado do *stream*
w_{pr}	Janela temporal do presente
w_{pa}	Janela temporal do passado
rt	Quantidade de eventos compartilhados
fc	Quantidade de *likes*
st	Similaridade entre os eventos
w	Matriz de convolução ou *kernel*
W_{xh}	Matriz de pesos conectando o i-ésimo neurônio da camada oculta aos neurônios de entrada
W_{hx}	Matriz dos pesos entre a camada oculta e a camada de saída da rede neural
h	Matriz de saída da camada oculta da rede neural
z	Matriz de saída da rede neural
e	Erro de reconstrução
b	Viés dos neurônios da camada oculta
S	Mapa de características ou mapa de ativação

k	Quantidade de vizinhos mais próximos
s	Pontuação (*score*) de anomalia ou função que retorna um *score* de anomalia
Q	Conjunto de atributos ou características de X
Q_j	Características aleatoriamente selecionadas sem reposição de X
d	Quantidade dimensões ou características $d =\mid Q \mid$
j	Uma única execução do algoritmo
OU	x considerado *outlier*
NC	x considerado *normal*
O_t	Algoritmo de detecção de *outlier*
l	Caminho possível na árvore *outlier*
ψ	Tamanho de uma subamostra
t	Número de árvores
T	Uma árvore ou um nó
$\phi(x)$	Retorna o tamanho do caminho de x na árvore
α(x)	Função de transformação não linear Sigmoid
$f(x)$	Função de transformação não linear ReLU
$\delta(x)$	Função de transformação não linear Softmax
V	Vértice de um grafo
E	Aresta de um grafo
$G(V, E)$	Estrutura de um grafo
μ	Quantidade de vértices de um grafo $\mu =\mid V \mid$
k_μ	Grafo completo

Sumário

1 INTRODUÇÃO 19
1.1 Objetivos 22
1.2 Questões norteadoras 23
1.3 Contribuições 23
1.4 Estrutura do trabalho 24

2 REVISÃO DA LITERATURA 25
2.1 Síntese da literatura 34
2.2 Proposta e motivação 36

3 REFERENCIAL TEÓRICO 37
3.1 Detecção de novidades 37
3.2 Fusão de dados 38
3.3 CNN 44
3.4 MASK R-CNN 48
3.5 Algoritmos não supervisionados 53
3.6 kNN 56
3.7 HBOS 57
3.8 *Feature Bagging* 59
3.9 *Isolation Foresting* 61
3.10 *Autoencoder* 64
3.11 *LSTM* 67
3.12 *Word embedding* ou vetores densos 69
3.13 Grafos 74

4 MÉTODO PROPOSTO 77
4.1 Definição de novidade 77
4.2 Qualidade da novidade 78
4.3 Heurística de rotulagem semiautomática 79
4.4 Codificação do texto e imagem 83
4.5 Fusão do texto e da imagem 88

5 EXPERIMENTOS 93
5.1 Métrica 93
5.2 Base de dados 95
5.3 Configuração dos experimentos 96

5.4 **Resultados** **97**
5.5 **Discussões** **110**

6 **CONCLUSÕES FINAIS** **114**
6.1 **Compêndio** **114**
6.2 **Trabalhos futuros** **115**

REFERÊNCIAS **117**

1 INTRODUÇÃO

Nos anos recentes foi testemunhado o desenvolvimento dos serviços de mídias sociais, que mudaram a forma como as pessoas viviam, trabalhavam e comunicavam. Especialmente, com o grande crescimento dos *smartphones*, um usuário acessa a internet a qualquer momento e em qualquer lugar, e compartilha por meio das redes sociais o que acontece ao seu redor. As redes sociais é um meio de transmissão que permite aos usuários trocar conteúdos multimídia. Esse conteúdo é heterogêneo, e inclui: texto, imagens, vídeos, informações temporais, conexões sociais, preferências dos usuários e outros metadados. Além disso, os usuários na *web* podem ver e comentar o conteúdo compartilhado no mundo inteiro.

Atualmente, uma diversidade de serviços de redes sociais, como Facebook, Twitter, Yelp, WebChat, Weibo e etc, se beneficiam em adquirir e espalhar informações. A popularidade vem de seus serviços diferenciados de comunicação, como portabilidade, rapidez e facilidade de uso, que permitem aos usuários responderem instantaneamente e disseminar informações com restrições limitadas ou nenhuma restrição de conteúdo. Hoje em dia, qualquer pessoa que testemunhe ou esteja envolvida em qualquer evento é capaz de disseminar informações em tempo real, que podem alcançar o outro lado do mundo à medida que o evento se desenrola (ATEFEH; KHREICH, 2015). Diante do grande volume de informações geradas é necessário identificar o que é de fato relevante e novo. A literatura denomina as seguintes técnicas utilizadas para detecção de dados relevantes e novos, como detecção de novidades ou detecção de evento (PIMENTEL et al., 2014).

A detecção de novidade é geralmente referenciada na literatura como detecção de evento (ATEFEH; KHREICH, 2015). Na verdade, detecção de evento é um termo geral usado para referenciar uma variedade de tipos de detecção incluindo *outlier*, anomalias, entre outros. Além disso, muitos termos sinônimos, como detecção de anomalias ou *outliers*, são frequentemente usados para se referir a detecção de novidade. Entretanto, há uma diferença entre significados. Em (MERRIAM-WEBSTER, 2018) novidade é definida como um evento que é "novo ou diferente de tudo que seja familiar". Em (BARNETT; LEWIS, 1994), um *outlier* é definido como um evento que é "inconsistente em relação ao conjunto dos dados de treinamento". Portanto, a detecção de *outlier* visa identificar observações anormais do conjunto de dados com o fim de descartá-las (PHAM et al., 2013). Enquanto que na tarefa de detecção de novidades, se uma novidade é encontrada, então será considerada como parte do conjunto de dados, em oposição a um *outlier*. Apesar disso, ambas as definições compartilham da mesma desvantagem, que é a escassez de amostras na fase de treinamento. Por esta razão, detecção de novidades ou *outliers* utilizam algoritmos

de classificação não-supervisionados ou *one-class*. Nos casos em que é possível ter amostras rotuladas na fase de treinamento, o problema é tratado como uma versão não balanceada de detecção de novidades (AGGARWAL, 2016).

A escassez de amostras acontece porque os dados anormais ou novos são caros de serem obtidos ou ocorrem em baixa frequência (PIMENTEL et al., 2014). Essas situações podem ocorrer nos seguintes contextos: um defeito em uma máquina; a identificação de um padrão de tráfico anômalo em uma rede de computadores, que pode significar um computador invadido por *hackers*; a detecção de uma imagem anômala de ressonância magnética, que pode indicar a presença de tumores malignos; a descoberta de anormalidades em dados de transação de cartão de crédito, que pode indicar furto do cartão; a detecção de leituras anômalas de um sensor, que pode significar alguma falha em algum componente mecânico; a revelação de fluxo temporal anormal de mensagens nas redes sociais, que pode caracterizar algum evento importante ou inesperado, etc. (CHANDOLA; BANERJEE; KUMAR, 2009).

Esses sistemas podem funcionar em operação normal por um grande período de tempo. Os dados gerados em operação usual são geralmente acessíveis e de fácil medição. Por outro lado, observar eventos de falhas ou raros depende da ocorrência de todos os possíveis cenários em operação imprópria, porém nem sempre é possível obter todos esses cenários. Ainda, o problema pode se tornar mais difícil se existir uma significante variabilidade entre os modos anormais, alguns dos quais podem não ser conhecidos a priori, o que torna a classificação supervisionada ou multiclasse inadequados para essas aplicações (PIMENTEL et al., 2014). A tarefa de detecção de novidades tem uma dificuldade inerente em obter amostras, porque as novidades são eventos que nunca aconteceram antes ou porque elas são muito raras.

No caso das redes sociais online (*microblogs*), milhares de notícias são geradas diariamente, cobrindo uma variedade significante de assuntos (IBRAHIM et al., 2018). A detecção automática de notícias que são novidades e de grande audiência, como desastres naturais ou um prêmio Nobel, se torna algo muito singular e desafiante para ser predito. Consequentemente, modelos supervisionados de classificação têm maior probabilidade de apresentarem alto viés enquanto mantém baixa variância (PIMENTEL et al., 2014). Uma das soluções para este problema é oferecida pelos algoritmos não supervisionados, no qual não é necessário possuir amostras dos *outliers* ou novidades. A descrição da normalidade é aprendida pela construção de um modelo com numerosos exemplos representando situações de normalidade (PIMENTEL et al., 2014).

Uma outra questão que surge no campo de detecção de novidades, é o tratamento de dados heterogêneos, pois trata-se de uma característica inerente das mídias sociais. As notícias postadas nas redes sociais podem conter não apenas texto mas também imagens, vídeos, informações temporais, conexões sociais, preferências dos usuários, e

outros metadados. Gao et al. (2015) mostraram que 30% dos blogs postam conteúdo com imagens, e este número continua a aumentar. A proliferação de aplicações que compartilham fotos na web, como Flickr, Instagram, Pinterest, e Vero, resultam em um grande número de imagens sendo disponibilizadas publicamente. O Instagram recebe 95 milhões de fotos e vídeos diariamente (LISTER, 2019). O Facebook é atualmente a rede social mais popular no compartilhamento de fotos, tendo cerca de 100 milhões de imagens compartilhadas diariamente (LA TIMES NEWS, 2018). Outra rede social que é conhecida pela publicação de curtas mensagens de textos (conhecidos como *tweets*) é o Twitter, com o total de 1,3 bilhões de contas. Há mais de 500 milhões de *tweets* postados em único dia, ou aproximadamente 6000 *tweets* por segundo. De acordo com Smith (2019), *tweets* com imagens resultam em uma taxa de cliques 18% maior, 89% mais de curtidas (*likes*), e 150% mais de compartilhamentos.

A literatura discute vários problemas encontrados quando imagens são classificadas da mídia social, incluindo baixa qualidade, significado ambíguo, ruídos, entre outros. Como consequência, a maioria das técnicas atuais de detecção de novidades para *web* utilizam conteúdo textual e conexões sociais, enquanto conteúdo visual é ignorado. Entretanto, o conteúdo textual postado nas redes sociais pode ter muitos problemas, incluindo abreviações, gírias, paráfrases, metáforas, alta expressividade, entre outros, que tornam difícil a tarefa de classificação (DODGE; KARAM, 2016) (MINER et al., 2012). Recentemente, o campo de visão computacional teve significativos avanços devido técnicas de *deep learning*, mais especificamente, redes neurais convolucionais profundas. Essas novas técnicas permitem extrair informações das imagens com excelente precisão. Exemplos incluem a identificação de diferentes objetos em uma cena, incluindo pessoas, animais, veículos, entre outros. Esse tipo de informação pode ser usada como complemento à informação do texto, para desenvolvimento de melhores classificadores de dados das redes sociais. Neste contexto, uma das hipóteses deste trabalho é que a fusão de texto e imagens adicionará mais informações relevantes a detecção de novidades em casos onde estruturas isoladas (textos ou imagens) podem ser deficientes.

Outra peculiaridade é que as mídias sociais geram dados de forma contínua, ou seja, no formato de *streaming*. A quantidade de dados produzidas nas mídias sociais não é apenas enorme mas continua a crescer, e os dados são regularmente atualizados (MOHAMMADI et al., 2018) (SHI et al., 2017). Devido a considerável quantidade de dados e o objetivo específico das maiorias das aplicações, muitos trabalhos tentam identificar novidades dentro de um contexto específico, algumas vezes filtram os dados por palavras chaves, como é feito em (ZHONG, 2018), (GHELANI et al., 2017), (AGGARWAL; SUBBIAN, 2012) e (SCHINAS et al., 2018). Como contribuição deste trabalho, o conceito de novidade foi generalizado a fim de torná-lo livre de um contexto específico. Dessa forma, a novidade é redefinida como um item singular que se destaca em relação às janelas temporais do presente e do passado.

Além disso, ao olhar as janelas do passado, atribui-se a ideia de memória. Por exemplo, seja a janela temporal de 1 dia e a notícia é "Cotação do dólar". Como a detecção de novidades se comportaria para esta curta janela temporal? Se fosse alterada a janela temporal de 1 dia para 20 dias, a detecção de novidade identificaria as mesmas notícias como novidade? A literatura mostra que uma mudança do tamanho na janela temporal muda o rótulo e, portanto, implica em uma mudança na distribuição fundamental do problema, que é conhecido na literatura como *concept drift* (WANG; MINKU; YAO, 2018) (MASUD et al., 2018). Isso evidencia outra característica peculiar da detecção de novidade que difere da detecção de *outlier*. Enquanto na detecção de novidades o aspecto temporal pode alterar a natureza do rótulo, na detecção de *outlier* o dado anômalo sempre será anômalo seja qual for a janela temporal.

O aspecto temporal é importante para identificação da novidade, mas é necessário caracterizar a importância ou relevância da novidade. Portanto, as interações dos usuários nas redes sociais nos dizem muito sobre o mundo real, e essas ideias não são limitadas a qualquer aspecto temporal particular (HENDRICKSON et al., 2019). Há muitas outras formas de caracterizar eventos de interesse dos usuários das mídias sociais. Talvez a forma mais simples é baseada nos conteúdos dos *tweets* (KHATER; GRACANIN; ELMONGUI, 2017). Isso significa que a identificação estaria ligada a um contexto, ignorando um evento importante de outro contexto. Uma outra forma, é identificar a popularidade do evento, o qual indica o grau de interesse dos usuários. Como Hong, Dan e Davison (2011) destacaram, a popularidade dos *tweets* é afetada por múltiplas razões, além da notoriedade. Portanto, neste estudo buscou-se caracterizar a novidade pelo grau de relevância (notoriedade), que é definido como a qualidade da novidade.

Por fim, este estudo abordará a detecção de novidades na rede social Twitter em três aspectos: heterogeneidade, volatilidade e audiência. A fusão de dados trata o aspecto da heterogeneidade dos dados, englobando texto e imagem na detecção. As janelas temporais são dirigidas ao aspecto de volatilidade ou temporal dos dados. A qualidade da novidade conduz o aspecto de audiência, ou seja, quão notório é a notícia na rede social.

1.1 Objetivos

O objetivo principal deste trabalho é desenvolver uma abordagem baseada em redes neurais profundas que realize detecção de novidades em dados de uma rede social online, no caso específico deste trabalho a rede Twitter. Além disso, pesquisar a inclusão de dados heterogêneos, tais como imagens e textos no sentido de tentar aprimorar a detecção.

Outros objetivos secundários também são traçados:

- Pesquisar a inclusão de dados heterogêneos, tais como imagens e textos no sentido

de tentar aprimorar a detecção de eventos.

- Criar uma base de dados heterogênea específica para detecção de novidades.
- Gerar um modelo matemático que caracterize a novidade em termos temporal e independente do contexto.
- Identificar a importância da novidade em termos de audiência.
- Avaliar métodos de fusão de dados.
- Avaliar técnicas técnicas não supervisionadas, inclusive de *deep learning*, que são pouco exploradas neste contexto.

1.2 Questões norteadoras

Esta seção diz respeito às questões que direcionarão a pesquisa e que demandarão ações condizentes com os objetivos predefinidos para que sejam encontradas respostas satisfatórias. Portanto, a pesquisa sugere responder as seguintes questões norteadoras:

- A fusão de dados heterogêneos, em especial os métodos avaliados, podem melhorar a detecção de novidades?
- O paradigma de janelas temporais permite a detecção de novidades livre de contexto?
- De que forma a qualificação da novidade utilizando o atributo de audiência pode contribuir na detecção de novidades?

1.3 Contribuições

A maior contribuição deste trabalho é a proposta de uma abordagem de detecção que utiliza a fusão de dados heterogêneos (imagens e textos) para detecção de novidades em mídias sociais, e que pode ser aplicada mesmo quando somente um dos tipos de dados está presente.

Três artigos foram publicados como resultado desta pesquisa:

- Amorim, Marta; Bortolotti, Frederico; Ciarelli, Patrick Marques; Salles, Evandro O. T. ; Cavalieri, Daniel C. . *"Novelty Detection in Social Media by Fusing Text and Image into a Single Structure"*. IEEE Access, v. 4, p. 1-1, 2019. (AMORIM et al., 2019)

- Amorim, Marta; Ciarelli, Patrick Marques. *"Fusão de redes convolucionais heterogêneas para detecção de novidades em redes sociais'*. In: CBA, 2018, João Pessoa. Congresso Brasileiro de Automática 2018, 2018. (AMORIM; CIARELLI, 2018)

- Amorim, Marta; Bortolotti, Frederico; Ciarelli, Patrick Marques; De Oliveira, Elias; De Souza, Alberto Ferreira. *"Analysing rotation-invariance of a log-polar transformation in convolutional neural networks"*. In: 2018 International Joint Conference on Neural Networks (IJCNN), 2018, Rio de Janeiro. (AMORIM et al., 2018)

1.4 Estrutura do trabalho

Este trabalho está organizado como segue.

No Capítulo 1 foi contextualizado o problema de detecção de novidades em redes sociais, apresentada uma breve revisão da literatura e as dificuldades inerentes ao problema a ser tratado neste trabalho. Por fim, são apresentadas as questões norteadoras que esta tese pretende responder.

No Capítulo 2 são apresentados alguns trabalhos relacionados a detecção de novidades e fusão de dados. Além disso, é feita uma revisão de literatura das principais técnicas a serem usadas neste trabalho.

O Capítulo 3 abrange os assuntos mais importantes e as bases fundamentais que serão tratados neste trabalho.

No Capítulo 4 são descritos os métodos propostos: o conceito de novidade, a qualidade da novidade, como o texto e a imagem são codificados em vetores, duas abordagens de fusão e quais algoritmos não supervisionados foram utilizados.

No Capítulo 5 são apresentadas as configurações dos experimentos, características do conjunto de dados, a heurística de rotulagem semiautomática do conjunto de dados, os resultados, discussões e análises.

Finalmente, no Capítulo 6 são realizadas as considerações finais e propostos os trabalhos futuros para continuação deste trabalho.

2 REVISÃO DA LITERATURA

Muitos trabalhos têm sido propostos para detecção de novidades em diversas aplicações. Neste capítulo será feita uma breve revisão da literatura focando nos trabalhos mais recentes aplicados a detecção de novidades usando dados não estruturados (imagem, texto, etc.).

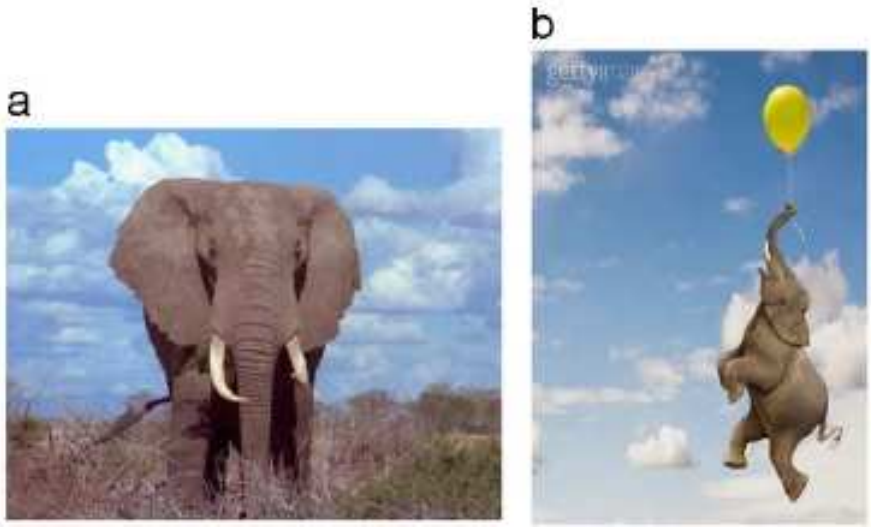
Phuvipadawat e Murata (2010) apresentaram um método para coletar, agrupar, classificar e rastrear novidades que sejam *breaking news* no Twitter. Primeiro são realizadas consultas predefinidas, por exemplo, palavras-chave: *"#breakingnews"* e *"#breaking news"*, e indexam o conteúdo. Mensagens que são semelhantes umas às outras são agrupadas para formar uma notícia. As mensagens são representadas como vetores e os pesos das palavras são dados pela método de TF-IDF (*Term Frequency–Inverse Document Frequency*) Manning, Raghavan e Schütze (2008), com um aumento de peso para nomes próprios, *hashtags* e nomes de usuários. Os nomes próprios são identificados usando o Reconhecedor de Entidades de Stanford (NER, 2018) treinado em corpus de notícias convencionais. A abordagem usa uma combinação ponderada pela quantidade de seguidores (confiabilidade) e a quantidade de mensagens compartilhadas (popularidade) mais recentes para identificar cada *cluster* como *breaking news*. Novas mensagens são incluídas em um *cluster* se são semelhantes à primeira mensagem e aos termos desse *cluster*. Os autores enfatizam a importância da identificação de nomes próprios para melhorar a comparação de similaridade entre *tweets* e, portanto, melhorar a precisão geral do sistema.

Aggarwal e Subbian (2012) propuseram um algoritmo de aprendizado *online* para tratar textos da rede social Twitter e e-mails. O algoritmo é composto de *clusters* que são continuamente atualizados com a chegada de novos dados. Para a medida de similaridade entre um evento e um *clusters*, as normas L1 e L2 são usadas. O conjunto de dados contém 517.432 e-mails e 1.628.779 *tweets*. Os rótulos das classes foram associados com *hashtags*: *tweets* foram rotulados em duas classes, e e-mails foram categorizados em 19 classes baseados nos *tokens* mais comuns. Na fase experimental, o algoritmo identificou corretamente 60% de todas as novidades.

Yong, Deng e Purvis (2012) propuseram um *framework* para detecção de novidade em imagens de animais selvagens. A novidade nesse contexto é identificada como algo não usual a vida selvagem, como no exemplo ilustrado na Figura 1. De forma geral, o funcionamento do *framework* possui o seguinte fluxo: segmentação da imagem, extração das características de cada bloco segmentado, rotulação do bloco, modelagem do contexto semântico e classificação da imagem. Para a segmentação da imagem foi escolhido o algoritmo JSEG (*J value segmentation*). Para a extração das características da imagem foi

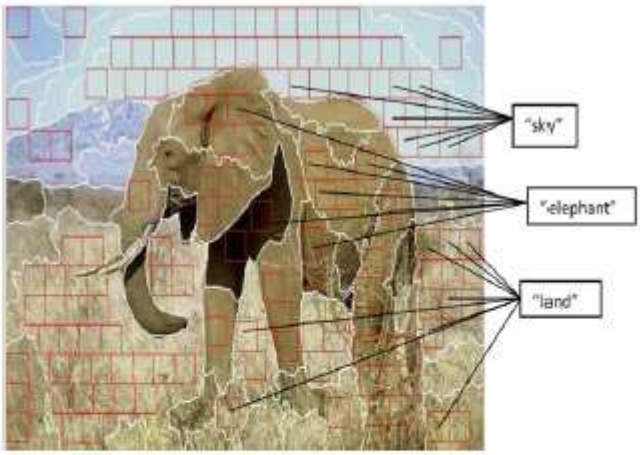
usado histograma de cores LUV, histograma de borda e filtro de Gabor com descritores de Haralick. A rotulação dos blocos é feita pela classificação de cada segmento da imagem utilizando o algoritmo 1-NN. Após a identificação dos rótulos que são objetos da imagem, uma matriz de ocorrências é gerada. Essa matriz esparsa representa o contexto semântico da imagem, pois relaciona todos os objetos da cena. A Figura 2 ilustra o processo anterior. Por fim, a imagem é identificada como normal ou novidade por meio de um método de classificação semissupervisionada. De forma objetiva, o funcionamento do algoritmo de classificação baseia-se na determinação do centroide calculado por meio das matrizes de ocorrência e calcula a distância entre a imagem e o centroide. Se a imagem possuir uma distância maior que um (*threshold*, o algoritmo o classifica como novidade. Caso contrário, a imagem é considerada normal. Para validar a proposta, um experimento foi conduzido com 335 imagens de animais selvagens extraídas do Google e Microsoft Research.

Figura 1 – A figura a) representa um imagem normal e b) representa uma novidade.



Autoria (YONG; DENG; PURVIS, 2012)

Figura 2 – Matriz de ocorrências.



class	coast	land	sky	water	dolphin	elephant	penguin	zebra
coast	0	0	0	0	0	0	0	0
land	0	1	0	0	0	1	0	0
sky	0	1	1	0	0	1	0	0
water	0	0	0	0	0	0	0	0
dolphin	0	0	0	0	0	0	0	0
elephant	0	1	1	0	0	1	0	0
penguin	0	0	0	0	0	0	0	0
zebra	0	0	0	0	0	0	0	0

Autoria (YONG; DENG; PURVIS, 2012)

Aiello et al. (2013) compararam seis métodos de detecção de eventos em três conjuntos de dados do Twitter relacionados a assuntos de esporte e política. Os métodos de detecção usados foram: LDA (*Latent Dirichlet Allocation*), Document-pivot, GFeat-p, FPM (*Frequent Pattern Mining*), SFPM (*Soft Frequent Pattern Mining*), BNgram. Os três conjuntos de dados possuem apenas texto e foram pré-processados usando as técnicas de tokenização e *stemming*. Os autores utilizaram técnicas de n-gram e TF-IDF para transformar o texto em vetores esparsos, e assim servirem de entrada para as técnicas de detecção. A intenção do trabalho também foi detectar eventos em janelas temporais, para isso se basearam em um cenário centrado no usuário. O usuário inicia o sistema de detecção fornecendo a semente temporal, a taxa de atualização e as palavras-chave. Os experimentos indicaram o BNgram como o melhor método para detecção de eventos.

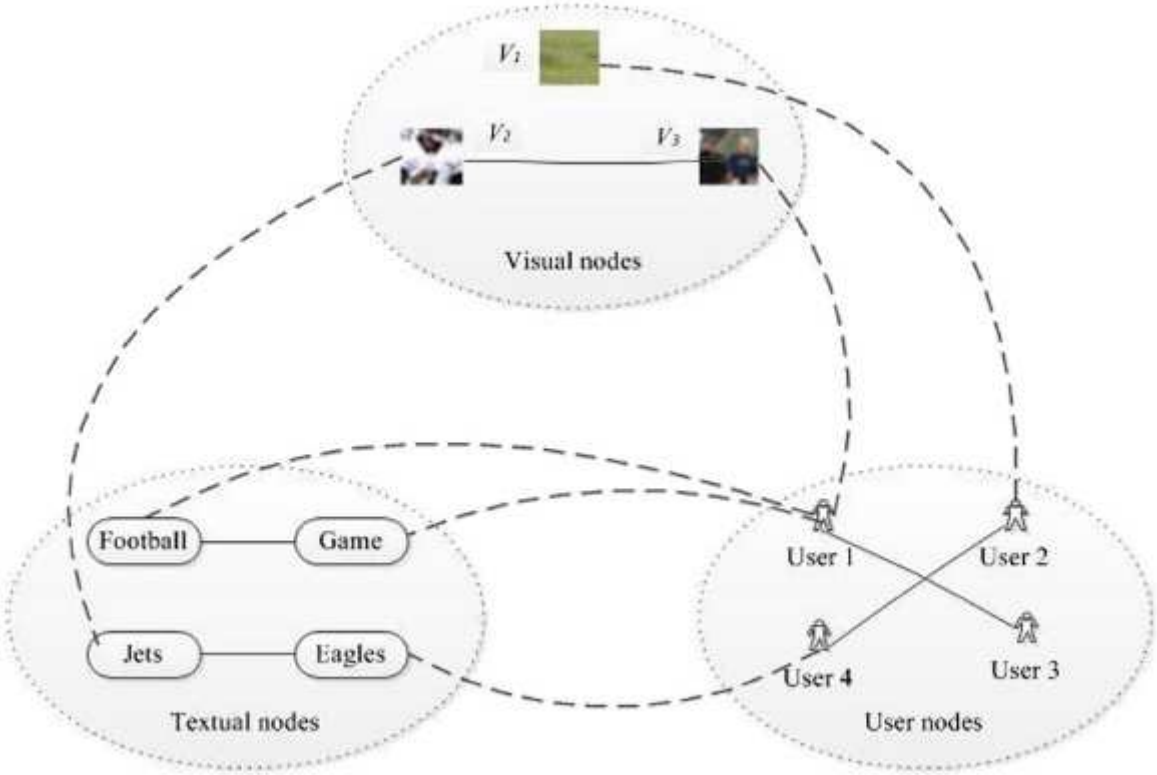
Por sua vez, Ding et al. (2015) consideram como novidades as intenções dos usuários das redes sociais em consumir ou adquirir algo. Para isto, os autores propuseram um modelo chamado *Consumption Intention Mining Model* (CIMM) baseado em uma rede neural convolucional para texto. Para demonstrar a efetividade do modelo, experimentos foram realizados em dois assuntos distintos: crianças e filmes. O Corpus textual contém 20 milhões de *posts*, 76 milhões de sentenças, e 1,3 bilhões de palavras. Os experimentos obtiveram acurácia de 94,52% para crianças e 85,28% para filmes.

Alqhtani, Luo e Regan (2015) desenvolveram um método para detectar eventos no Twitter. O objetivo é avaliar se a detecção de eventos é melhor usando apenas texto, ou apenas imagem, ou com a fusão dos dois. O método monitora os *streams* de dados e seleciona os *tweets* que tem textos e imagens e os armazenam em uma base de dados. Isso é seguido por algumas técnicas de pré-processamento de texto, tais como: filtro dos caracteres não pertencentes ao idioma inglês, tokenização, remoção de *stopwords* e aplicação de *stemming*. O procedimento inicia com a detecção do evento baseado apenas no texto, usando o modelo de *bag-of-words* e a ponderação de TF-IDF. Então é realizada a detecção do evento baseado apenas na imagem, usando as características visuais, como HOG (*Histogram of Oriented Gradients*), GLCM (*Gray-Level Co-Occurrence Matrix*) e histograma de cores. A partir dessas características visuais, é realizada uma classificação com kNN (*k Nearest Neighbors*) que categoriza cada novo caso usando a distância Euclidiana. Para avaliar a técnica foi realizado um experimento com um milhão de *tweets* postados no dia 28 de dezembro de 2014. O resultado do experimento mostrou que o método proposto atingiu a acurácia de 89% com texto, 86% com imagem e 94% com os dois.

Zhang et al. (2016) propuseram um novo modelo probabilístico que identifica o evento por meio da correlação de vários tipos de dados, como: conteúdo visual, textual, perfil do usuário e informação temporal. O conteúdo textual inclui: *tags*, títulos, descrições e comentários das imagens; e é denotado por um vetor $x_t =< x_{t1}, x_{t2}, \ldots, x_{ti} >$, sendo x_{ti} um valor baseado na frequência do i-ésimo termo do instante t. O conteúdo visual adotado foi baseado em características visuais, ou seja, palavras que representam o conteúdo visual. A imagem é dividida uniformemente em pedaços de 16×16. As características visuais são extraídas em torno de cada *keypoint* identificado por SIFT (*Scale-Invariant Feature Transform*) (GONZALEZ; WOODS, 2018) e por meio de histogramas de cores de 32 e 128 dimensões. Após isso, as características são agrupadas e convertidas em palavras visuais com auxílio do algoritmo *k-means* (AGGARWAL; REDDY, 2013). Assim, o conteúdo visual pode ser representado por $x_v =< x_{v1}, x_{v2}, \ldots, x_{vi} >$, sendo cada elemento associado a uma palavra visual v. Já as características do usuário estão relacionadas a imagem, ou seja, o usuário que fez o *upload* da imagem, usuários que compartilham imagem ou os que marcam como favorita. Dado um usuário u, as características deste usuário são denotadas por $x_u =< x_{u1}, x_{u2}, \ldots, x_{ui} >$. Um evento C é algo que ocorre em um certo instante de tempo, e que geralmente é representando por um conjunto de conteúdo multimídia. Um conjunto k de eventos pode ocorrer em um instante de tempo t e podem ser categorizados em um conjunto $E = \{C_i | i = 1, 2, \ldots, k\}$. Cada evento C_i pode ser considerado como um "objeto virtual" que é composto pela união das características $C_i =< \cup x_{tj}, \cup x_{vj}, \cup x_{uj} > (j = 1, 2, \ldots, |C_i|)$. A correlação das características são estruturadas em um modelo de grafo, conforme ilustrado na Figura 3, em que as características são os vértices e as arestas são as relações entre as características. Foi empregado o modelo probabilístico de Markov para conectar cada nova característica com o evento correto. Os experimentos foram realizados com dois

milhões de imagens extraídas do Flickr.

Figura 3 – Grafo de correlação entre os tipos de dados.



Autoria (ZHANG et al., 2016)

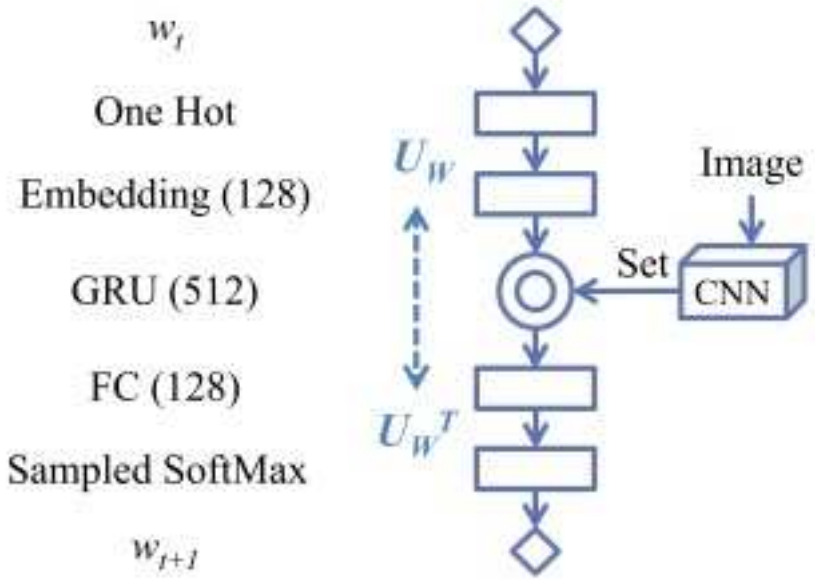
Kaneko e Yanai (2016) apresentam um sistema para descobrir eventos relacionados a fotos do Twitter. As fotos selecionadas incluem localização geográfica (latitude/longitude). O método consiste de dois principais passos: detecção das palavras-chave baseada na análise textual e no *clustering*. Primeiro, a detecção do evento é feita apenas com a análise textual. Após isso, as características visuais são extraídas das fotos. Para a extração de características das imagens, foram utilizados *bag-of-features* (BoF), SURF (*Speeded Up Robust Features*) (GONZALEZ; WOODS, 2018) e histograma de cores RGB (*Red-Green-Blue*) de 64 dimensões. As fotos são agrupadas nas palavras-chave correspondentes, usando o método de *clustering* Ward, que é um método de agrupamento hierárquico, e a foto mais representativa do evento é selecionada. Nessa fase foi usado o algoritmo *VisualRank* para a seleção da foto de maior representatividade do evento. Um diferencial nessa abordagem foi na seleção amostral das palavras-chave pela frequência diária dentro de uma unidade de área. Uma área é uma unidade delimitada por latitude e longitude no mapa para detectar um evento. Foi utilizado o mapa dos Estados Unidos e Japão. Isso é importante, pois a frequência diária das palavras-chave aumentam muito mais em algumas unidades no mapa do que em outras. Por exemplo, a atividade dos usuários do Twitter em cidades grandes como Nova York e Tokyo é muito maior que em outras cidades como Idaho e Fukushima.

Os experimentos analisaram 17 milhões de *tweets* geolocalizados postados nos Estados Unidos e 3 milhões postados no Japão.

O objetivo central do trabalho de Mao et al. (2016) foi treinar e avaliar *word-embeddings* eficazes que continham em sua codificação tanto informação textual quanto visual. Por isso, propuseram uma rede neural, conforme Figura 4, que utilizou como entrada dados heterogêneos (textos e imagens) das redes sociais para gerar *word-embeddings* que representassem adequadamente as frases. Apenas textos são transformados em *embeddings*, enquanto as informações das imagens são extraídas por meio de uma rede CNN. A rede neural utiliza uma arquitetura recorrente (RNNs) inspirada no modelo baseado em legenda de imagem CNN-RNN (VINYALS et al., 2014) (MAO et al., 2015). A representação visual foi mapeada no mesmo espaço dos estados GRUs (*Gated Recurrent Unit*) (ou seja foi configurado $h_o = ReLU(W_I f_I)$). Uma vez que a informação visual é alimentada depois da camada *embedding*, geralmente é dificil assegurar que esta informação é misturada em *embedding* aprendidos. Por isso, foi adotada a estratégia de peso transposto compartilhado proposto em (MAO et al., 2015) que foi usado para o desafio do modelo aprender novos conceitos visuais. Mais especificamente, a matriz de pesos da camada SoftMax U_M é compartilhada com a matriz U_W da camada de *embedding* de maneira transposta. A camada U_M se refere a matriz de pesos da rede CNN. Dessa maneira, U_W^T é aprendida a decodificar a informação visual e incorpora essa informação na matriz de *word-embedding*.

Também foi construído uma base de dados com 300 milhões de frases e 40 milhões de imagens extraídas da rede social Pinterest. Os usuários do Pinterest geralmente marcam imagens com descrições curtas e compartilham as imagens com outras pessoas. Como frases e imagens podem ser compartilhadas e marcadas por vários usuários, muitas imagens têm um conjunto rico de descrições. O modelo treinado extrai *word-embeddings* para todas as palavras e usa a média dos vetores para representação das frases. Em seguida, é verificada se a distância do cosseno entre o par (frase base, frase positiva) são menores que a (frase base, frase negativa), por exemplo: frase base (*hair style*), frase positiva (*ponytail*) e frase negativa (*pink nail*). Na avaliação, são usadas triplas de frases rotuladas, em que cada tripla contém frase A, frase B, frase C. A frase A é considerada semanticamente mais próxima de B do que A é de C. No final, a distância entre o *embedding* $A - B$ e $A - C$ são calculados, e é considerada uma tripla positiva se $d(A, B) < d(A, C)$. Esta abordagem de comparação é comumente usada para avaliar e comparar diferentes modelos de *word-embeddings*. A conclusão dos experimentos é que a fusão de texto e informação visual ajuda significantemente o aprendizado de *word-embedding*. O modelo proposto supera em 9.5% o modelo de *word2vec*.

Figura 4 – Modelo de rede neural para dados heterogêneos geral.



Autoria (MAO et al., 2016)

Qian et al. (2018) abordam uma nova técnica de fusão de dados heterogêneos das redes sociais (imagem, texto e usuários) no processo de sumarização de eventos. Primeiro, foram extraídas as características das imagens usando o SIFT (*Scale Invariant Feature Transform*). Cada uma das 128 dimensões do SIFT é quantizada para um vocabulário visual pela hierarquia de quantização (QIAN et al., 2015). Então, cada imagem é representada por uma dimensão de 10.000 *bag of word* visuais. No segundo passo, um vetor de alta dimensão é calculado usando *bag of words* do texto. As similaridades entre os vetores de textos e os vetores de imagens são calculados usando a distância do cosseno. Por fim, é calculada uma matriz de correlação entre as imagens e os textos usando as similaridades anteriormente calculadas, que gera uma pontuação de relevância entre o texto e a imagem. O conjunto de dados usados nos experimentos foram extraídos da rede social Weibo contendo cerca de 2.776.623 textos e 4.432.514 imagens. Apenas 38,71% dos textos têm imagens vinculadas.

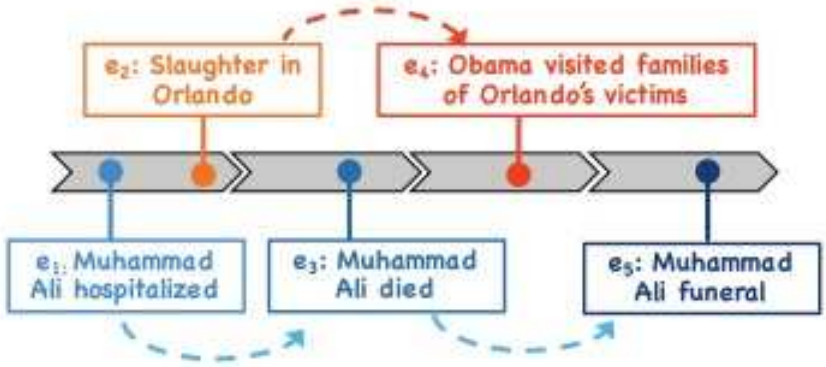
No trabalho de Mei et al. (2018) foi aplicada uma combinação de algoritmos de *clustering* e *autoencoder* para detectar novidades em *Brainstorming*. O aspecto desafiante dos dados consistiu em sentenças breves com 5 a 12 palavras, tornando a análise muito difícil. O conjunto de dados tem 2.583 documentos, cada sentença sendo uma ideia individual. Os resultados foram avaliados usando as métricas MCN (*Mean Cluster Novelty*) e HFN (*High Novelty Fraction*), com MCN igual 0,5577 para *clusters* de tamanho 10, e 0,4272 para *clusters* de tamanho 50.

Considerando *cyberbulling* como uma novidade, Rosa et al. (2018) implementaram uma CNN (KIM, 2014), uma C-LSTM Híbrida (ZHOU et al., 2015), e NN-LSTM-DNN

(GHOSH; VEALE, 2016). A C-LSTM Híbrida é formada por uma CNN que extrai representações de alto-nível de frases e alimentam uma rede LSTM. Os autores a nomearam de C-LSTM porque apresentou melhores resultados que as redes CNN e LSTM. A C-LSTM também mostrou bons resultados nas tarefas de classificação de sentimentos e classificação do tipo de pergunta. A rede NN-LSTM-DNN usou uma abordagem similar, e é formada por 2 camadas de CNN seguida de 2 camadas de LSTM e finalmente, uma camada de rede neural densa (DNN), atingindo o estado da arte em dois diferentes conjuntos de dados para detecção de sarcasmo. A CNN de Kim (2014) foi projetada para a tarefa de classificação binária orientada a sentimentos. A CNN consiste de uma camada convolucional seguida por uma camada totalmente conectada com dropout de 0,5 e saída Softmax. Para testar as três redes de Kim (2014), Zhou et al. (2015) e Ghosh e Veale (2016), os pesquisadores usaram uma versão extendida do conjunto de dados da Formspring, disponível na Kaggle com o total de 13.160 textos. O melhor resultado foi atingindo usando o Google *embeddings* com a CNN (KIM, 2014).

O trabalho de Mele, Bahrainian e Crestani (2019) realiza detecção, rastreamento e previsão de eventos de vários fluxos de notícias. O trabalho propõe uma abordagem denominada *dDTM-News* (*Dynamic Topic Model News Streams*) que recebe como entrada documentos provenientes de múltiplos fluxos (blogs, mídias sociais, etc.) e é capaz (i) de descobrir os tópicos latentes discutidos nos documentos (detecção de eventos) e (ii) rastrear sua evolução ao longo do tempo (rastreamento de eventos). O *dDTM-News* primeiro divide o conjunto de dados em intervalos de tempo e aplica a técnica LDA (*Latent Dirichlet Allocation*) para extrair os assuntos latentes. Após isso, os assuntos são encadeados sobre o tempo usando HMM (*Hidden Markov Model*), conforme ilustrado na Figura 5. O objetivo é estimar a probabilidade de transição entre cada um dos estados (ou seja evolução dos assuntos). Para realizar a predição de eventos no fluxo temporal é utilizado o filtro de Kalman. Foi construído um conjunto de dados de documentos de 9 canais de 3 plataformas diferentes (Twitter, portais de notícias RSS, sites de notícias) para um total de 27 tópicos de notícias. A coleção abrange 4 meses (de 1 de março a 30 de junho de 2016) e consiste em cerca de 147 mil documentos de notícias. Ao final, o algoritmo *dDTM-News* criou *clusters* baseados em eventos de novos documentos e comparou com as seguintes técnicas: *k-means*, *k-means* temporal e hLDA (*hierarchical LDA*). A técnica proposta pelos autores atingiu melhor *Recall* médio (0,87).

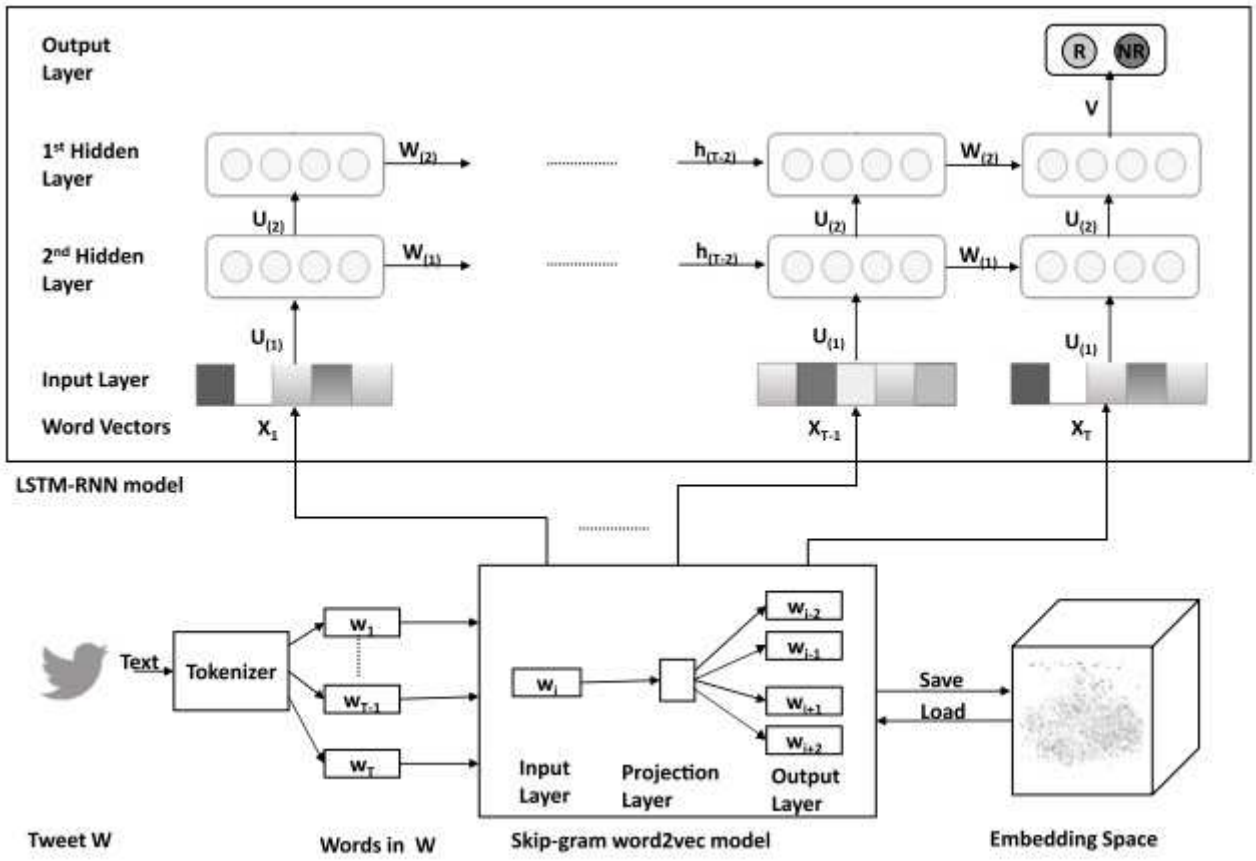
Figura 5 – Encadeamento de eventos sobre o tempo.



Autoria (MELE; BAHRAINIAN; CRESTANI, 2019)

O trabalho de Alkhodair et al. (2019) detecta um tipo particular de rumor que é semelhante a novidade: *breaking news*. Existem dois tipos de rumores ou boatos na mídia social basedos nas características temporais: *long-standing* e *breaking news*. Os *long-standing* são discutidos por um longo período de tempo e podem facilmente ter dados coletados para treinamento. Em contraste, rumores *breaking news* geralmente não possuem observações para treinamento e ocorrem em tempo real (ZUBIAGA et al., 2018). O autor propõe um modelo *word2vec* treinado não supervisionado para aprender os vetores representativos das palavras e treina uma rede LSTM-RNN supervisionada para detecção do rumor. A Figura 6 ilustra a arquitetura do modelo. O conjunto de dados usados nos experimentos contém 4.842 *tweets* com 9 eventos. Cada rumor é anotado como sendo Verdadeiro, Falso ou Não Verificado. Os experimentos utilizaram os seguintes algoritmos para comparação com o proposto: SVM, *Naive Bayes*, *Random Forest*, Entropia Máxima e *Conditional Random Field*. O melhor resultado foi para o algoritmo proposto com F1 igual a 0,795. O segundo melhor resultado para o algoritmo *Random Field* com F1 igual a 0,761. Uma limitação apontada pelo autor é que um *tweet* emergente que foi sinalizado como boato pode mais tarde ser não-boato. No entanto, o modelo proposto não modela ou memoriza os fatos ao longo do tempo.

Figura 6 – O modelo proposto por Alkhodair et al. (2019) para detectar rumor. Um tweet w é primeiro tokenizado em uma sequência de palavras $w =< w_1, \ldots, w_T >$. Próximo passo, o modelo word2vec converte a sequência de palavras em sequência de vetores $x =< x_1, \ldots, x_T >$, que são entradas da rede LSTM-RNN. Finalmente, a rede LSTM-RNN prediz a classe de saída no tempo T.



Autoria (ALKHODAIR et al., 2019)

2.1 Síntese da literatura

A Tabela 1 resume os métodos relatados na revisão da literatura. A coluna "Resultado" informa o melhor resultado alcançado entre as bases de dados testadas.

Tabela 1 – Lista de métodos relatados na revisão da literatura

Autor	**Base de dados**	**Técnicas de Detecção**	**Tipo**	**Resultado**
Aggarwal e Subbian (2012)	517.432 emails 1.628.779 *tweets*	Agrupamento online	Texto	0,3 precisão/0,6 recall
Yong, Deng e Purvis (2012)	335 fotos	Algoritmo 1-NN	Imagem	0,89 ROC-AUC
Aiello et al. (2013)	-	LDA, Dpivot, GFeat-p, FPM , SFPM, BNgram	Texto	0,7692 recall
Ding et al. (2015)	20 milhões de mensagens	CNN	Texto	0,945 recall
Alqhtani, Luo e Regan (2015)	1 milhão de *tweets* postado no dia 28 de dezembro de 2014.	kNN	Texto e Imagem	0,94 acurácia
Zhang et al. (2016)	2 milhões de imagens associadas a outras informações (textuais,temporal, etc)	Naïve Bayes para texto k-Means e SVM para imagem.	Texto, Imagem Usuário	0,77 precisão/0,88 recall
Kaneko e Yanai (2016)	17 milhões de *tweets* geolocalizados postados nos Estados Unidos e 3 milhões no Japão	Algoritmo não supervisionado	Texto e Imagem	0,889 precisão
Mao et al. (2016)	154 mil *tweets*	Agrupamento e Ranqueamento	Texto	0,843 precisão
Qian et al. (2018)	300 milhões frases e 40 milhões imagens	RNN	Texto e Imagem	0,843 precisão
Mei et al. (2018)	2.583 documentos	Agrupamento e Autoencoder	Texto	0,5577 MCN
Rosa et al. (2018)	13.160 textos	CNN	Texto	0,848 *F-Measure*
Mele, Bahrainian e Crestani (2019)	147 mil documentos	HMM, LDA, kNN e Kalman	Texto	0,83 *F-Measure*
Alkhodair et al. (2019)	4.842 *tweets*	Rede LSTM-RNN	Texto	0,795 *F-Measure*

2.2 Proposta e motivação

A partir da pesquisa bibliográfica, pode-se evidenciar algumas fraquezas e oportunidades que podem ser pontos de aprofundamento desta pesquisa:

- Os algoritmos da literatura, em sua grande maioria, buscam identificar os eventos usando apenas textos ou apenas imagem. Ambos os tipos de dados são conhecidos na literatura por possuírem desafios que devem ser enfrentados pelos algoritmos. No caso dos textos, problemas como: paráfrases, metáforas, sentido conotativo, erros de grafia, variedade linguística, etc. No caso das imagens, problemas como: baixa qualidade, sentido ambíguo, ruído, iluminação, perspectiva de visão, etc.
- A maior parte das abordagens tratam novidades em contextos específicos.
- Grande parte dos trabalhos da literatura identificam novidades sem levar em consideração a natureza temporal da novidade.
- Poucos trabalhos apresentam um *framework* completo para detectar novidades em fluxo contínuo de dados.
- Muitos trabalham com vetores esparsos para caracterizar o texto ou a imagem.
- Muitas bases de dados não estão publicamente disponíveis.

Dentro desta conjuntura, a hipótese central deste trabalho é que a fusão do texto e imagem adicionará novas informações, principalmente se as informações forem extraídas de forma eficaz dos dados. Além disso, é proposto um novo arcabouço completo para tratar novidades independente do contexto e, de forma geral, segue os seguintes passos: receber *streams* de dados heterogêneos (imagem ou texto) da mídia social, realizar a fusão da imagem ao texto, codificar a imagem ou texto e por fim utilizar algoritmo de aprendizado para identificar a novidade.

3 REFERENCIAL TEÓRICO

3.1 Detecção de novidades

A detecção de novidades, algumas vezes mencionada na literatura como detecção de eventos, tem sido abordada há muito tempo no programa TDT (*Topic Detection and Tracking*) (SCHANDA; SANDERSON; CLOUGH, 2014), uma iniciativa patrocinada pela Agência de Projetos de Pesquisa Avançada de Defesa Americana (DARPA), preocupada com a organização baseada em eventos de fluxos de notícias textuais. A motivação para a iniciativa de pesquisa da TDT foi fornecer tecnologia básica para ferramentas de monitoramento de notícias a partir de múltiplas fontes de mídia tradicional (incluindo *newswire* e *broadcast* de notícias) para manter os usuários atualizados sobre notícias. Originalmente, o TDT consistia em três tarefas principais: segmentação, detecção e rastreamento. Essas tarefas tentam segmentar o texto da notícia em histórias coesas, detectar eventos novos (imprevistos) e acompanhar o desenvolvimento de um evento relatado (ATEFEH; KHREICH, 2015).

De acordo com a TDT, o objetivo da detecção de eventos é descobrir eventos novos ou não identificados anteriormente, onde cada evento se refere a uma coisa específica que acontece em um momento e lugar específico (ALLAN et al., 1998) (ATEFEH; KHREICH, 2015). Nesse contexto da TDT, a detecção de eventos foi amplamente dividida em duas categorias: detecção de eventos retrospectivos (RED) e detecção de eventos novos (NED) (ALLAN et al., 1998) (YANG; PIERCE; CARBONELL, 1998). O RED concentra-se em descobrir eventos previamente não identificados de coleções históricas acumuladas (YANG; PIERCE; CARBONELL, 1998), enquanto o NED envolve a descoberta de eventos novos em fluxo contínuo de dados (ALLAN et al., 1998).

A partir de então, muitas técnicas de detecção de novidades sugiram e foram aplicadas em distintos contextos. Estes incluem a detecção em mamografias (TARASSENKO et al., 1995) e outros problemas de diagnóstico médico (QUINN; WILLIAMS, 2007) (CLIFTON et al., 2011), em sistemas industriais complexos (PIMENTEL et al., 2014), sistemas de detecção de intrusão, tais como detecção de fraudes em cartões de crédito ou telefones celulares (PATCHA; PARK, 2007), vigilância por vídeo (DIEHL; HAMPSHIRE, 2002), robótica móvel (SOFMAN et al., 2011), redes de sensores (ZHANG; MERATNIA; HAVINGA, 2010) e mineração de texto (BASU; BILENKO; MOONEY, 2004).

No *survey* de Pimentel et al. (2014) foram detalhados vários algoritmos do estado da arte para detecção de novidades e eles apresentaram uma taxonomia com a seguinte

classificação: (i) métodos probabilísticos, (ii) métodos baseados em distância, (iii) métodos baseados em reconstrução, (iv) métodos baseados no domínio, e (v) métodos baseados em teoria da informação. A abordagem (i) usa métodos probabilísticos quando frequentemente envolve estimativa de densidade das classes. Esses métodos assumem que áreas de baixa densidade no conjunto de treinamento indicam que têm uma baixa probabilidade de conter dados normais. A abordagem (ii) inclui os conceitos de vizinho mais próximo e análise de agrupamento que também são usados em problemas de classificação. A suposição aqui é que os dados normais estão fortemente agrupados, enquanto os novos dados ocorrem longe de seus vizinhos mais próximos. A abordagem (iii) envolve o treinamento de um modelo de regressão. Quando os dados são mapeados usando o modelo treinado, o erro de reconstrução entre o valor real observado e o valor alvo, dá origem a um *score* em que, dependendo do contexto e *threshold*, pode ser definido o que é ou não novidade. Abordagem (iv) usa métodos baseados em domínio para caracterizar os dados de treinamento. Esses métodos geralmente tentam descrever um domínio contendo dados e definindo um limite em torno da classe novidade, a qual segue uma distribuição dos dados. Abordagem (v) calcula usando medidas teóricas da informação a partir dos dados de treinamento, como entropia ou complexidade de Kolmogorov. O principal conceito aqui é que novos dados alteram significativamente o conteúdo da informação em um conjunto de dados.

Gupta et al. (2014) realizaram uma compilação de métodos de detecção de *outliers* para dados temporais e que apresentam algumas restrições essenciais que podem ser arbitrariamente combinadas, dependendo do problema, por exemplo, dados de séries temporais versus multidimensionais, ponto versus janela e o tipo de dados. O trabalho visa organizar em uma taxonomia baseado no tipo de dados, como séries continuas (ex: leitura de sensores), séries discretas (ex: web logs), fluxo de dados dimensionais (ex: fluxo de dados textual) e dados de redes (ex: grafos e *stream* de dados), que requerem tipos diferentes de métodos e análises. Gupta et al. (2014) enfatizam que técnicas baseadas em janelas podem ser melhores para identificar anomalias comparadas às técnicas que analisam toda série temporal diretamente.

3.2 Fusão de dados

Na literatura de aprendizado supervisionado, duas abordagens de fusões podem ser encontradas: fusão na entrada e na saída dos algoritmos. A última retorna à combinação das probabilidades da saída dos classificadores. Exemplos disso são as abordagens: *Random Forest*, *Bagging*, *Bragging*, *Wagging*, *Subagging*, *Rotated Bagging*, etc. (AGGARWAL; SATHE, 2017).

A fusão na saída dos classificadores, também conhecida na literatura como *ensemble* de classificadores, é usada para melhorar o desempenho do classificador instável ou fraco. A

instabilidade é no sentido que pequenas perturbações no conjunto de dados de treinamento podem resultar em muitos erros de predição (BREIMAN, 1999).

O princípio fundamental de um modelo *ensemble* é que um grupo de classificadores fracos juntos possam formar um classificador forte, que aumenta a acurácia do modelo. Quando é realizada alguma predição por qualquer algoritmo, a principal causa de diferença entre os valores *ground-truth* e preditos são: os ruídos, a variância e o viés. A fusão reduz esses fatores (exceto os ruídos que é um erro irredutível)(JAMES et al., 2014).

Os termos viés e variância vêm de uma conhecida decomposição do erro de previsão na regressão e foi originalmente desenvolvido para o erro quadrático. Vários autores propuseram outros tipos de decomposições, mas cada uma tem deficiências. Contudo, todas essas decomposições têm uma relação intuitiva com o erro quadrático (JAMES et al., 2014).

Seja um conjunto de treinamento $T = \{(y_n, x_n), \quad n = 1, ..., N\}$, sendo y_n as variáveis respostas e x_n as variáveis explicativas, algum modelo (redes neurais, árvores, regressão linear, etc) é aplicado ao conjunto para construir um preditor $f(x, T)$ que mapeie o valor x para y. Foi assumido que o conjunto de treinamento T consiste de amostras iid (independente e identicamente distribuídas) a partir da distribuição das variáveis $\{X, Y\}$ e que amostras futuras serão extraídas da mesma distribuição. Assume-se que existe uma relação entre as duas variáveis, conforme a Equação 3.1.

$$y = f(x) + e \tag{3.1}$$

Sendo e o erro normalmente distribuído com média zero. Supondo que seja obtido um modelo $\hat{f(x)}$ de $f(x)$ usando regressão linear ou qualquer outra técnica de modelagem. Portanto, o erro quadrático esperado no ponto x é descrito conforme Equação 3.2.

$$Err(x) = E[(y - \hat{f}(x))^2] \tag{3.2}$$

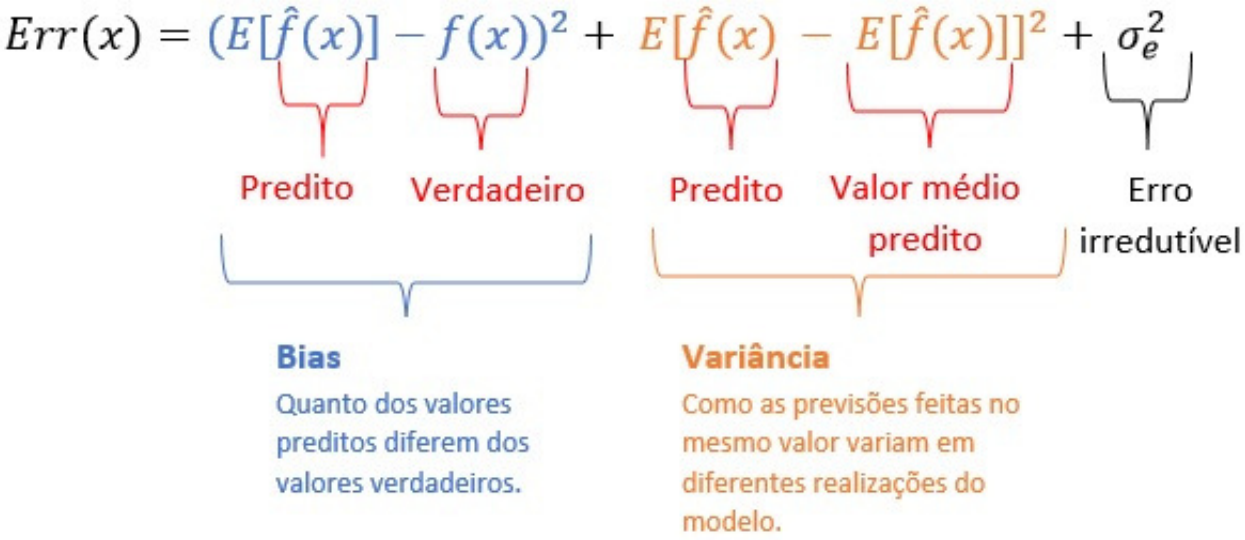
O erro $Err(x)$ pode ser ainda decomposto nas Equações 3.3 e 3.4.

$$Err(x) = (E[\hat{f}(x)] - f(x))^2 + E[(\hat{f}(x) - E[\hat{f}(x)])^2] + \sigma^2{}_e \tag{3.3}$$

$$Err(x) = Vies^2 + Variância + ErroIrredutível \tag{3.4}$$

O erro irredutível da Equação 3.4 é o erro que não pode ser reduzido. Por melhor que seja o modelo, os dados terão uma certa quantidade de ruído ou erro irredutível que não pode ser removido. A Equação 3.4 é conhecida como decomposição fundamental. A Figura 7 explica em mais detalhadamente o que foi apresentado na Equação 3.4:

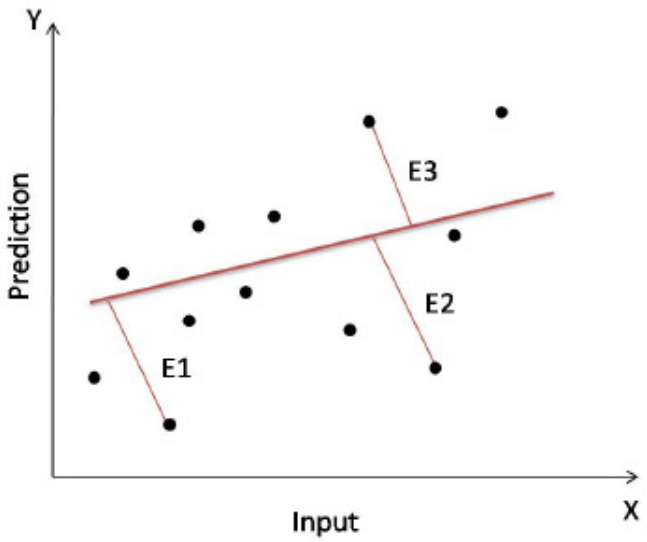
Figura 7 – Explicação gráfica da decomposição fundamental do erro.



Autoria (PATEL, 2019)

O *bias* ou viés é a média da diferença entre o resultado predito e o atual. Viés alto significa menor acurácia. Quando se treina um modelo linear com dados complexos é possível ver claramente erros, como demonstrado na Figura 8, em que os pontos pretos são os valores preditos e as linhas vermelhas representam a distância dos erros $E1$, $E2$ e $E3$. O significado é que o modelo não se ajustou apropriadamente. Para evitar esse cenário de *underfitting* é indicado usar um modelo mais complexo (polinomial) (PATEL, 2019).

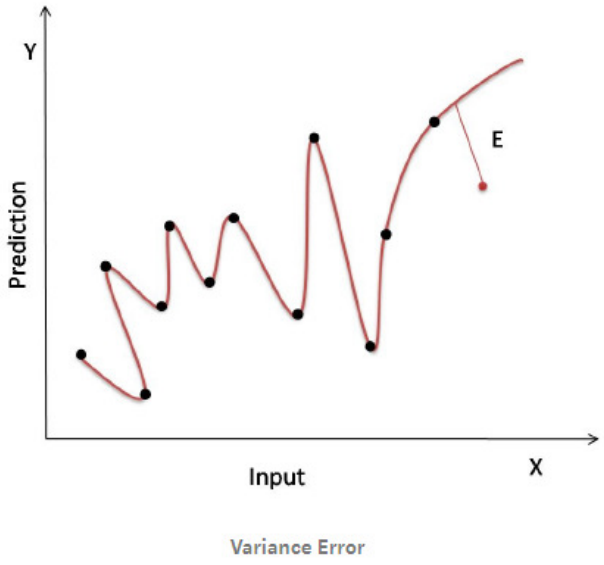
Figura 8 – Erro do tipo viés.



Autoria (PATEL, 2019)

A variância quantifica a diferença de um valor predito em uma mesma observação, ou seja, quanto cada valor predito irá variar em relação a outro valor. Modelos que mostram variância alta costumam apresentar bons desempenhos na fase de treinamento. No entanto, quando se realiza a fase de teste, o modelo falha bastante em predizer corretamente. Isso ocorre por dois principais motivos: 1) Poucos dados na fase de treinamento. 2) Modelo mais complexo do que o necessário para modelar os dados. A Figura 9, exemplifica um modelo bem ajustado aos exemplos de treinamento (pontos pretos). Mas quando o modelo tentar predizer (ponto vermelho) é possível observar o erro de variância. Esse problema de *overfitting* pode ser resolvido aumentando o número de amostras de treinamento ou escolhendo um modelo correto para a predição (PATEL, 2019).

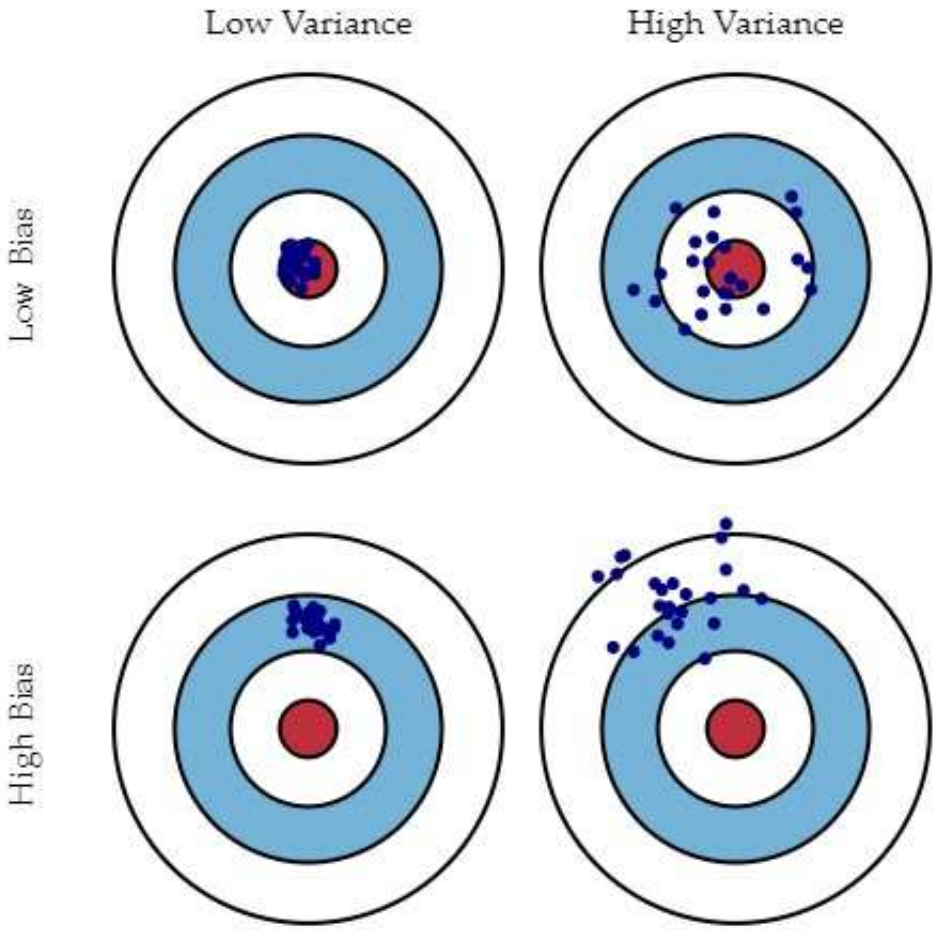
Figura 9 – Erro do tipo variância.



Autoria (PATEL, 2019)

Portanto, modelos com alto viés prestam muito pouca atenção aos dados de treinamento e simplificam demais o modelo. Sempre geram altos erros nos dados de treinamento e teste. A variância é a variabilidade da previsão do modelo para um determinado conjunto de dados. O modelo com alta variância presta muita atenção às amostras de treinamento e não generaliza para amostras que não haviam sido vistas antes. Como resultado, esses modelos têm um desempenho muito bom nas amostras de treinamento, mas apresentam altas taxas de erro nas amostras de teste. Há um *tradeoff* entre a habilidade do modelo minimizar o viés e a variância. A Figura 10 ilustra diferentes casos representando alto e baixo viés/variância (JAMES et al., 2014) (Scott Fortmann-Roe, 2018).

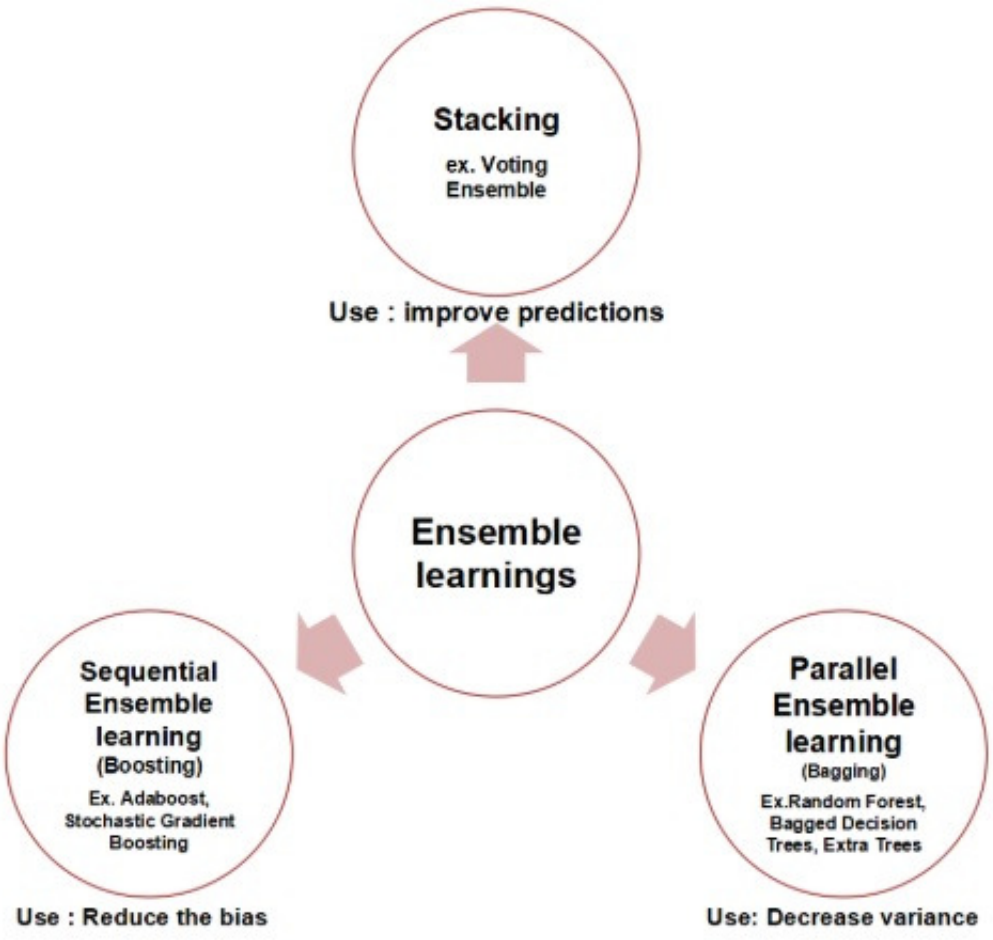
Figura 10 – Ilustração gráfica do viés e variância.



Autoria (Scott Fortmann-Roe, 2018)

A literatura tem demonstrado que o uso de *ensembles* tem ajudado a reduzir a variância e o viés. No entanto, há vários tipos de arquitetura de *ensembles*, que influenciam mais fortemente na variância que no viés, outras arquiteturas influenciam mais fortemente no viés e outros tentam alcançar um *tradeoff* entre viés e variância com o fim de melhorar as predições. A Figura 11 ilustra os tipos de arquiteturas mais utilizados, que são: *Bagging*, *Boosting* e *Stacking*. O *Boosting* atua principalmente em algoritmos supervisionados com foco na redução do viés, alguns exemplos de algoritmos que implementam a arquitetura são: *Adaboost* e *Stochastic Gradient Boosting*. Já o *Bagging* atua mais na redução da variância, alguns exemplos de algoritmos que implementam a arquitetura são: *Random Forest*, *Bagged Decision Trees* e *Extra Trees*. O *Stacking* é um forma de combinar múltiplos modelos, de maneira que as predições melhorem e reduza tanto o viés como a variância.

Figura 11 – Tipos de *ensembles*.



Autoria (PATEL, 2019)

Asmita e Shukla (2014) fizeram uma extensa revisão da literatura com diversos tipos de arquitetura de *ensembles*, como: usando *fuzzy*, algoritmos genéticos, *Borda Count* e *Feature bagging*. A conclusão da pesquisa é que a escolha da arquitetura do *ensemble* pode influenciar na redução do viés ou da variância.

No entanto, não há garantia de que a combinação de múltiplos algoritmos sempre funcionará melhor do que o melhor algoritmo individual (AGGARWAL; SATHE, 2017). Foi demonstrado que métodos de fusão (por exemplo *bagging*) não melhoram o desempenho de classificadores locais (por exemplo kNN) devido às predições correlacionadas entre os resultados de múltiplos classificadores combinados (BREIMAN, 1996a) (KONG; DIETTERICH, 1995a) e sua baixa sensibilidade à perturbação de dados (LAZAREVIC; KUMAR, 2005) (AGGARWAL; SATHE, 2017).

Outra questão importante é que algoritmos de detecção de novidades são problemas não supervisionados, e muitos dos algoritmos de fusão na saída usam rótulos em seu processo analítico. Por exemplo, métodos como *boosting* usam rótulos para avaliar a

acurácia dos dados e atribuir pesos. Esses métodos não podem ser generalizados para as técnicas de detecção de novidades por causa da indisponibilidade dos *ground-truth* (AGGARWAL; SATHE, 2015) (AGGARWAL; SATHE, 2017).

A fim de contornar os problemas da fusão na saída dos classificadores, têm surgido algumas abordagens de fusão na entrada. Essa fusão transforma as diversas entradas dos algoritmos em um único formato de representação comum. Essa abordagem tem sido bastante explorada quando os dados são provenientes de sensores, tais como apresentado em (AUDEBERT; SAUX; LEFÈVRE, 2017) e (SUN et al., 2018). Contudo, existem poucas pesquisas com fusão de dados heterogêneos na entrada, principalmente aplicadas para as redes sociais.

3.3 CNN

Rede Neural Convolucional (do inglês, *Convolutional Neural Networks* - CNN) é um tipo especializado de rede neural que emprega a operação matemática chamada de convolução. Inicialmente era amplamente utilizada apenas para tarefas de reconhecimento de objetos, mas hoje em dia está sendo usada em outros domínios, bem como no rastreamento de objetos, estimação de pose, detecção e reconhecimento de texto, detecção de saliência visual, reconhecimento de ação, rotulação de cena e muitos outros (ALOYSIUS; GEETHA, 2017).

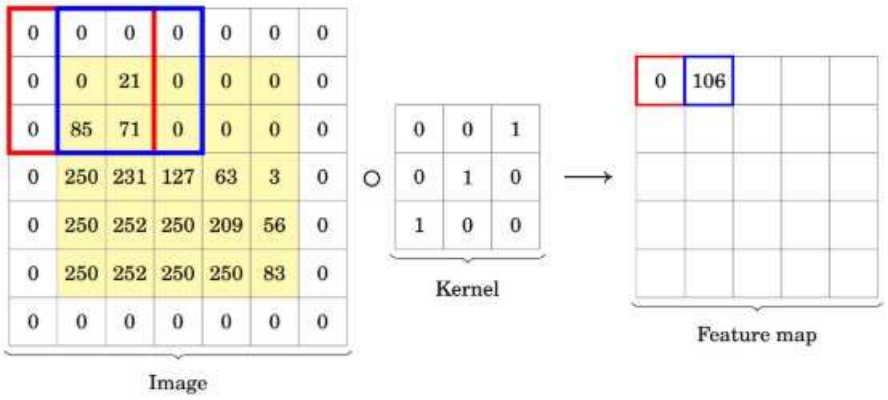
A rede neural Neocognitron (FUKUSHIMA, 1980) é considerada como a predecessora das CNN. A LeNet foi o trabalho pioneiro em CNN por LeCun et al. (1990), mas depois surgiram melhorias. Em 2012, Krizhevsky, Sutskever e Hinton (2012a) criaram um novo modelo CNN que conseguiu reduzir a taxa de erro na competição ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*). Ao longo dos anos, seu trabalho se tornou um dois mais influentes no campo da visão computacional e inspirou muitas outras variações de arquiteturas CNN. AlexNet conseguiu resultados notáveis em comparação com os modelos de aprendizado anteriormente utilizados. A arquitetura pode ser considerada como uma variante maior da LeNet, tendo cinco camadas convolucionais seguidas por três camadas totalmente conectadas. Houve várias variações da AlexNet desde seu enorme sucesso na competição ILSVRC-2012 (ALOYSIUS; GEETHA, 2017).

O tipo de camada da CNN na qual a maioria da computação ocorre é na camada convolucional. A convolução, em um sistema discreto, é um operador linear que, a partir de duas funções, resulta em uma terceira que mede a soma do produto dessas funções ao longo da região subentendida pela superposição delas em função do deslocamento existente entre elas. No contexto de imagens, a convolução é 2-D e é realizada entre uma imagem g e um filtro (kernel) f de dimensões $(2v + 1) \times (2v + 1)$, e a saída da operação é uma imagem h, conforme ilustrado na Equação 3.5 e na Figura 12 (GOODFELLOW; YOSHUA;

COURVILLE, 2016):

$$h[m,n] = \sum_{j=-v}^{v} \sum_{i=-v}^{v} f[i,j] \cdot g[m-i, n-j] \tag{3.5}$$

Figura 12 – Exemplo da operação de convolução 2-D.



Autoria (MIYAZAKI, 2017)

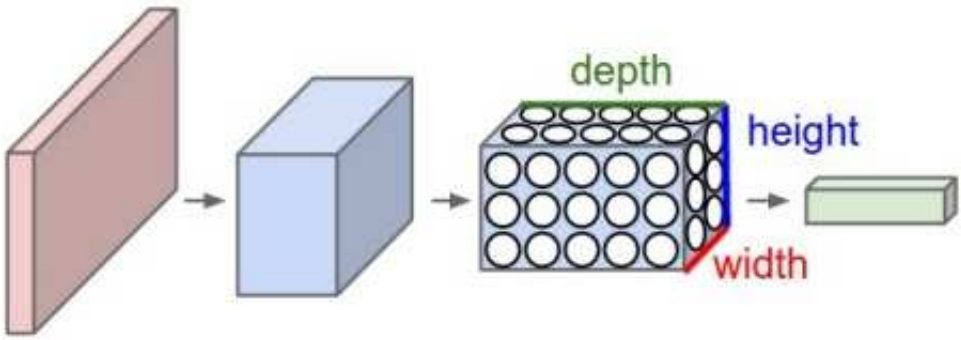
No contexto das redes neurais convolucionais, a convolução ocorre entre tensores (matrizes de dimensões 3-D ou 4-D), mais especificamente entre uma imagem (ou a saída de uma camada anterior) e um kernel, cujo valores de seus elementos foram aprendidos pela rede durante o treinamento. A saída desta operação é chamado de mapa de características (GOODFELLOW; YOSHUA; COURVILLE, 2016).

CNN são muito similar as redes neurais normais que podem ser visualizadas como uma coleção de neurônios organizadas como um grafo acíclico. A principal diferença da rede neural é que o neurônio da camada oculta é conectado apenas a um subconjunto de neurônios na camada anterior. Por causa dessa conectividade esparsa, é capaz de aprender recursos implicitamente. A arquitetura profunda da rede resultam em extração de características hierárquicas, ou seja, os filtros treinados da primeira camada contém características como bordas, da segunda contém formas, a próxima camada de filtros podem aprender partes dos objetos e os filtros finais podem identificar os objetos. As arquiteturas de aprendizado profundo podem possuir várias tipos de camadas especializadas para tentar extrair diferentes características, sendo as principais: camada convolucional, camada de *pooling*, camada totalmente conectada e camada de *Loss* (GOODFELLOW; YOSHUA; COURVILLE, 2016).

- Camada convolucional: Esta camada executa operações de convoluções na entrada para produzir um conjunto de ativações lineares, formando um conjunto de mapas de características como saída. Cada ativação linear pode ser passada por uma função de ativação não linear, como a função de ativação ReLU (*Rectified Linear Units*). Os

parâmetros aprendidos nesta camada são um conjunto de filtros ou *kernels*. Esses filtros são convoluídos com a entrada da camada (que pode ser uma imagem ou saída de uma camada anterior) para produzir mapas de ativação com 3 dimensões para imagens 2D ou mapas de ativação com 4 dimensões para imagens 3D, que quando empilhados ao longo da dimensão de profundidade, produz a saída. A Figura 13 demonstra a forma como estes mapas são empilhados e como cada camada aumenta sua profundidade em uma CNN com imagem 2D de entrada e matrizes 3D (comumente conhecido como tensor) nas camadas subsequentes (MIYAZAKI, 2017). Durante o processo de treinamento, esses filtros são ajustados automaticamente para que sejam ativados na presença de características relevantes, como orientação de bordas ou manchas de cores. Ainda observando a Figura 13, o primeiro elemento é a imagem, e segundo são os filtros, o terceiro é a saída. Cada camada convolucional varia a largura (*width*) e a altura (*height*) e a profundidade (depth) conforme os dados percorrem a CNN. Essa variação ocorre com a mudança de três parâmetros na operação de convolução: o movimento (passos) dos filtros (*stride*), o número de filtros em cada camada que altera a profundidade e o preenchimento (*padding*) na camada que sofrerá a convolução (GOODFELLOW; YOSHUA; COURVILLE, 2016).

Figura 13 – Camadas convolucionais.



Autoria (Karpath, A., 2019)

- Camada de *pooling*: A arquitetura CNN tem camadas de convolução alternando com camadas de *pooling* para reduzir a dimensão espacial dos mapas (sem perda de informação relevante) e o número de parâmetros em uma rede, tal que diminua a complexidade computacional. Esta abordagem controla o problema de *overfitting*. Uma função de *pooling* substitui a saída da rede em um determinado local por uma estatística resumida das saídas próximas. Por exemplo, o *maxpooling* (ZHOU; CHELLAPPA, 1988) obtém o valor máximo dentro de uma vizinhança retangular. Algumas das operações comuns de *pooling* são: *maxpooling*, *average pooling*, *stochastic pooling*, *spectral pooling*, *spatial pyramind pooling* e *multi-scale orderless pooling*. O trabalho de Springenberg et al. (2015) questiona a necessidade de usar o *pooling*, e

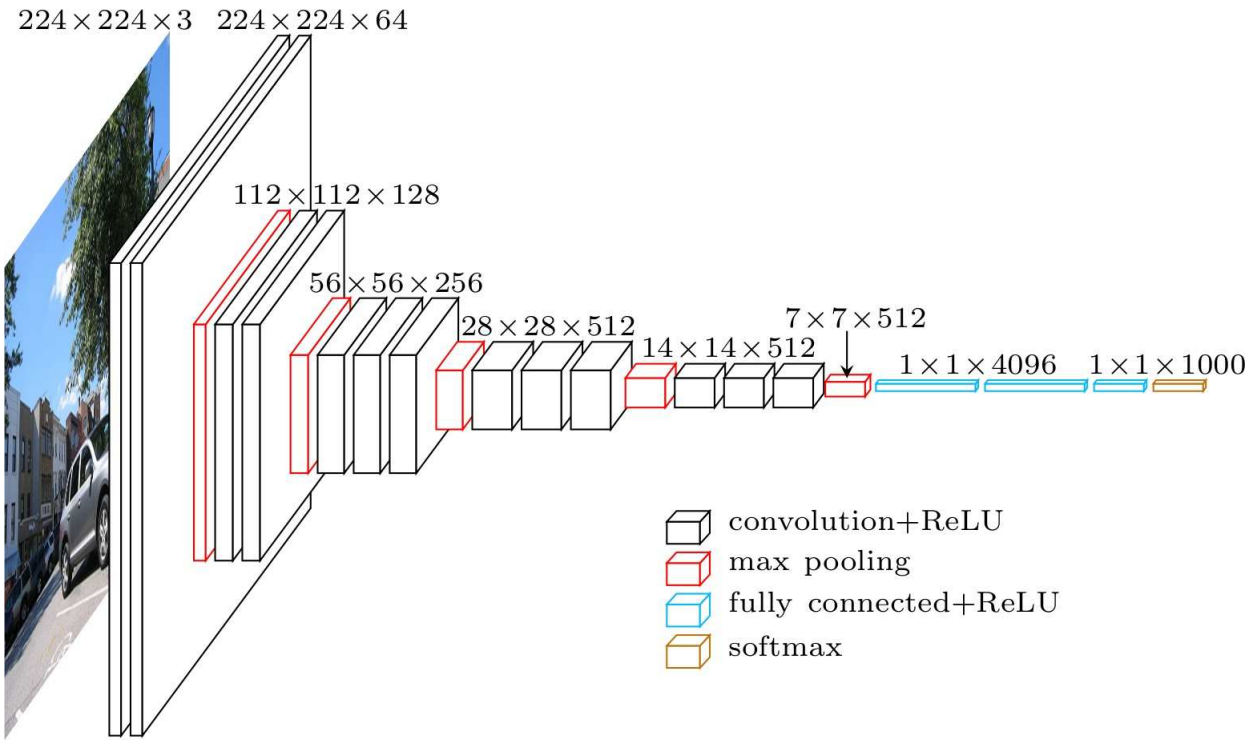
que poderia ser substituído por camadas convolucionais com *stride* igual a dois. Isto é principalmente aplicável em redes simples que provaram ter um desempenho superior a arquiteturas complexas (GOODFELLOW; YOSHUA; COURVILLE, 2016).

- Camada totalmente conectada: Os neurônios nesta camada são totalmente conectados a todos os neurônios das camadas anteriores, como uma rede neural clássica. Camadas deste tipo ocorrem nas últimas camadas de uma CNN. O raciocínio de alto nível é feito aqui. Os dados de entrada e saída destas camadas estão organizados na forma unidimensional (*flattened*), e não no formato de matrizes ou tensores, portanto não pode haver uma camada de convolução após uma camada totalmente conectada. Algumas arquiteturas tiveram sua camada totalmente conectada substituída, como apresentado por Zeiler e Fergus (2014), em que a camada totalmente conectada é substituída por uma camada de *average pooling* (GOODFELLOW; YOSHUA; COURVILLE, 2016).

- Camada de *Loss*: A última camada totalmente conectada serve como a camada que calcula a perda ou erro, que é uma penalidade por discrepância entre a saída desejada e a real. Para prever uma única classe de k classes mutuamente exclusivas, a função Softmax é usada (basicamente é a regressão logística multinomial). A função mapeia as previsões para valores não negativos e também é normalizado para obter probabilidade por classes (ASMITA; SHUKLA, 2014) (ALOYSIUS; GEETHA, 2017).

A Figura 14 mostra um exemplo de uma arquitetura convolucional (*Faster* R-CNN) composta de várias camadas (MIYAZAKI, 2017).

Figura 14 – Arquitetura de uma rede neural convolucional (*Faster* R-CNN).



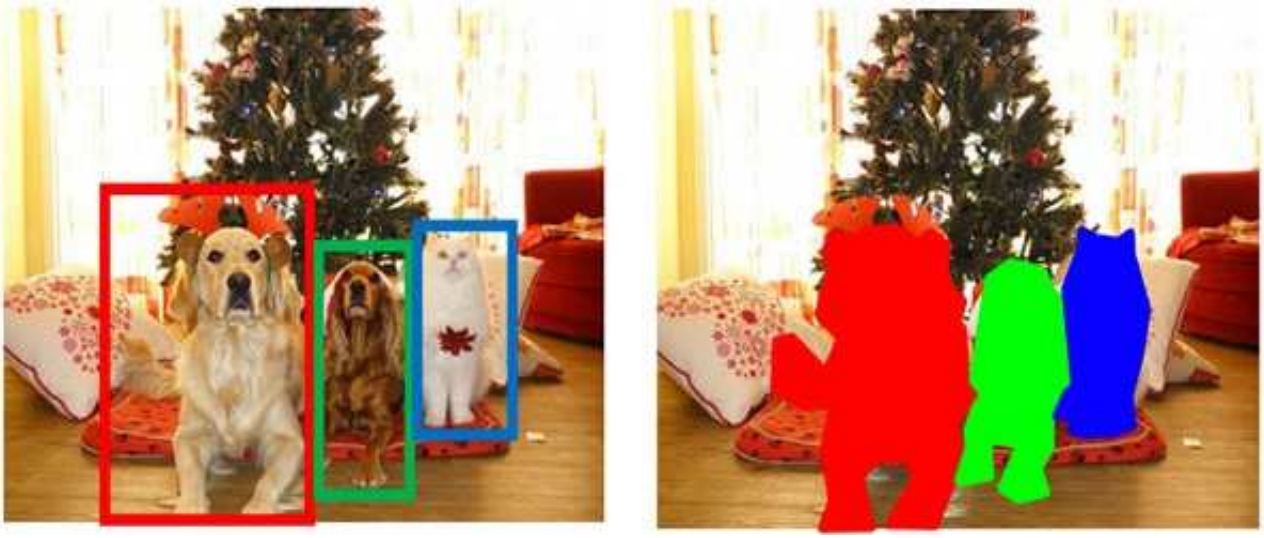
Autoria (Rey, J., 2019)

3.4 MASK R-CNN

A *Mask Region-based CNN* é uma rede convolucional cujo objetivo é realizar a segmentação de objetos em imagens. A *Mask R-CNN* é basicamente uma extensão da rede *Faster R-CNN*, que é largamente usada na tarefa de detecção de objetos. Entretanto, existem diferenças entre segmentar e detectar objetos em imagens (HE et al., 2017).

A detecção de objetos utiliza um *bounding box* correspondente a cada classe em uma imagem. Entretanto, não informa nada sobre a forma do objeto, pois apenas descreve as coordenadas do *bounding box*. A segmentação cria uma máscara de pixel para cada objeto. Essa técnica dá uma compreensão muito mais granular do objeto na imagem. A Figura 15 exemplifica a diferença entre detecção e segmentação de imagens (HE et al., 2017) (Karpath, A., 2019).

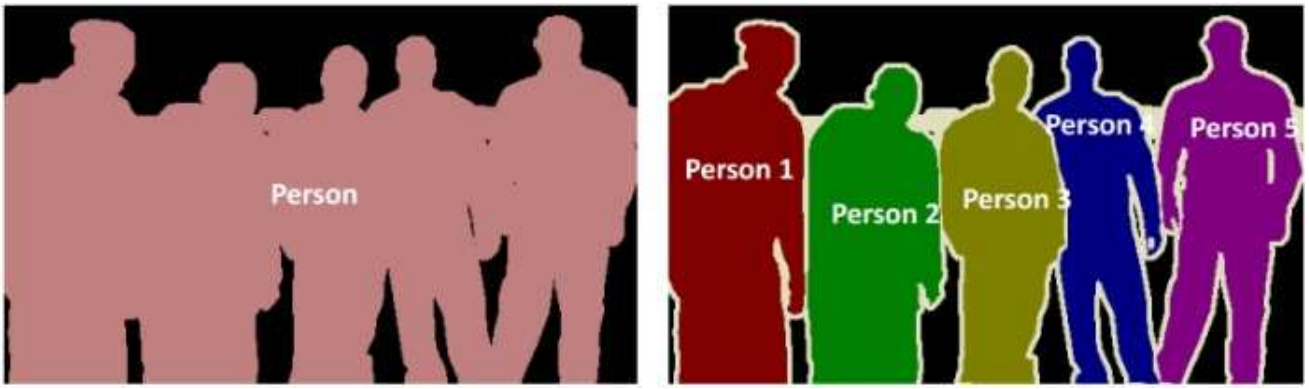
Figura 15 – A figura da esquerda mostra o resultado da técnica de detecção de objetos e a figura da direita mostra o resultado da técnica de segmentação de imagens.



Autoria (Karpath, A., 2019)

Há dois tipos de segmentação de imagens: segmentação semântica e a de instâncias. A Figura 16 mostra um exemplo para um melhor entendimento da diferença entre as duas técnicas. Todos os cinco objetos da Figura 16 são pessoas. A segmentação semântica classificará todas as pessoas como uma única instância. Entretanto, se os objetos forem da mesma classe, a segmentação de instâncias atribuirá como pessoas diferentes. O estado da arte para segmentação de instâncias utiliza na rede *MASK R-CNN* (HE et al., 2017).

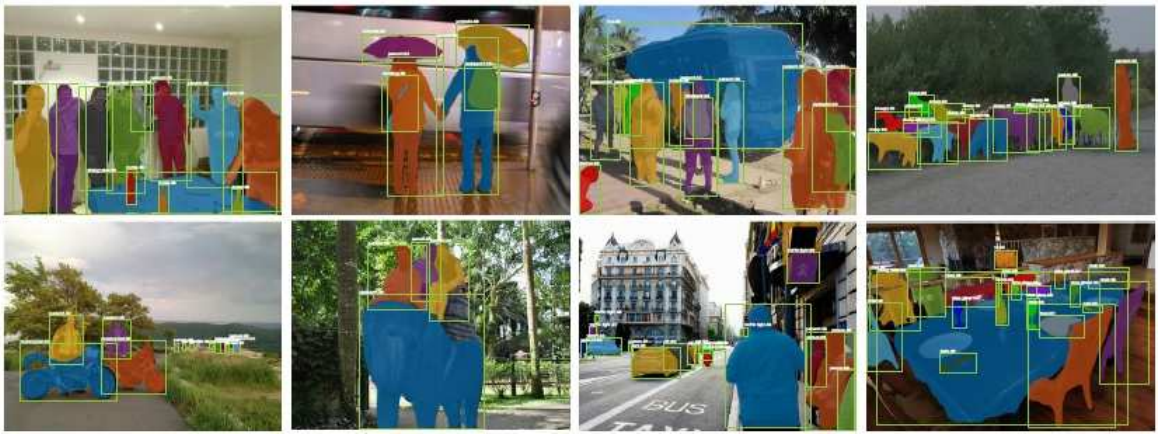
Figura 16 – A figura da esquerda mostra a técnica de segmentação semântica e a figura da direita mostra a técnica de segmentação de instâncias.



Autoria (Karpath, A., 2019)

Como a *MASK R-CNN* é construída baseada na *Faster R-CNN*, então para uma determinada imagem, além do rótulo da classe e das coordenadas dos *bounding box*, também é retornada a máscara de cada objeto em nível de pixel (HE et al., 2017). A Figura 17 ilustra um exemplo de imagens retornadas pela rede *MASK* R-CNN.

Figura 17 – Máscaras coloridas, *bounding boxes* e as probabilidades de confiança geradas pela *Mask R-CNN*.



Autoria (HE et al., 2017)

Para um entendimento geral da rede *MASK R-CNN*, será apresentado em passos o funcionamento:

1. Primeiro, a rede usa a arquitetura ResNet 101 para extrair mapas de características das imagens. A ResNet é uma rede convolucional com 101 camadas, que foi treinada para classificar imagens em 1000 categorias distintas.

2. Esses mapas de características são passados para *Region Proposal Network* (RPN) que retornam *bounding boxs* candidatos. RPN são algoritmos que aprendem dos mapas de características e propõem *bounding boxes*.

3. Os *bounding boxes* candidatos ou regiões preditas obtidas pelo RPN podem ser de diferentes tamanhos portanto, é aplicada uma camada de *RoI pooling* para que todos fiquem do mesmo tamanho.

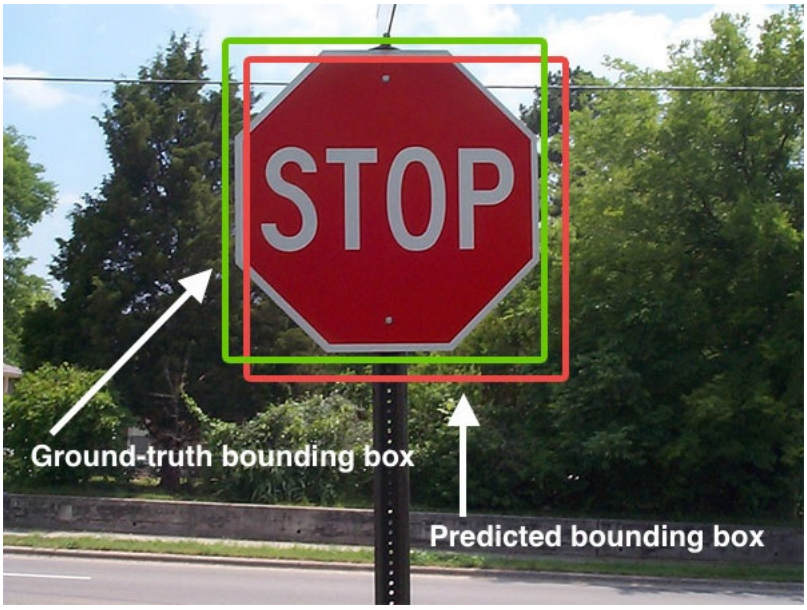
   Até este ponto, todos os passos funcionam de maneira similar a rede *Faster R-CNN*. O diferencial da rede *MASK R-CNN* é gerar a máscara de segmentação. Para isso, primeiro é calculada a região de interesse (*Region of Interest - RoI*). Portanto, para todas as regiões preditas, é calculado IoU (*Intersection over Union*) com os *bounding boxes ground-truth*. O IoU pode ser definido pela Equação 3.6:

$$IoU = AreaDeIntersecao/AreaDeUniao \tag{3.6}$$

   Para entender melhor como são extraídos os valores das variáveis $AreaDeIntersecao$ e $AreaDeUniao$, observe as Figuras 18 e 19. Nota-se que o classificador detectou a presença de um sinal de trânsito na Figura 18. O *bounding box* predito é traçado em vermelho enquanto o *bounding box ground-truth* é traçado em verde. A

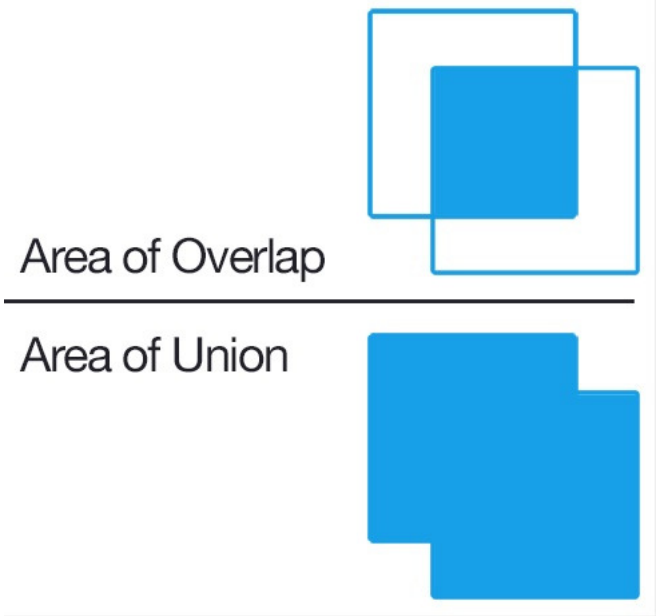
AreaDeIntersecao é a sobreposição entre o *bounding box* predito e o *bounding box ground-truth*. A *AreaDeUniao* é a área envolvida por ambos *boundings*. A divisão destas áreas produzem o *IoU*.

Figura 18 – Um exemplo de detecção de sinal de trânsito. O *bounding box* predito é traçado em vermelho enquanto o *bounding box ground-truth* é traçado em verde. O objetivo é calcular a área de interseção sobre a união destes *bounding boxes*.



Autoria (ROSEBROCK, 2016)

Figura 19 – O cálculo da interseção sobre a união é uma simples divisão das áreas dos *bounding boxes*.



Autoria (ROSEBROCK, 2016)

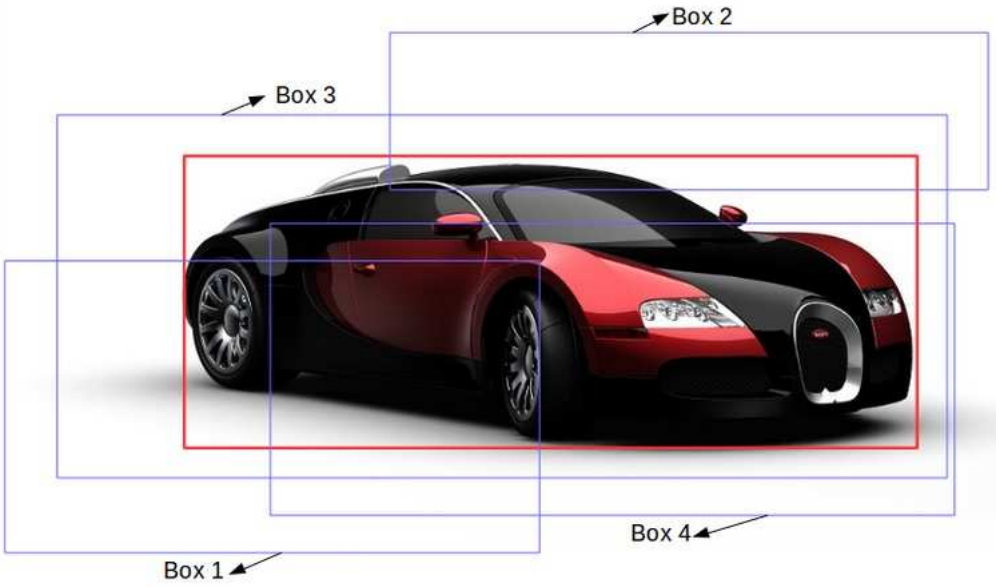
Se a *IoU* for maior ou igual a 0,5, é considerada como uma região de interesse. Caso contrário, é negligenciada essa região em particular. Isso é feito para todas as regiões e, em seguida, é selecionada apenas um conjunto de regiões para as quais a *IoU* é maior que 0,5. Para melhor entendimento, considere a Figura 20:

Figura 20 – Imagem com o *bounding box ground-truth.*

Autoria (ANALYTICSVIDHYA.COM, 2020)

O retângulo vermelho é o *bounding box ground-truth* da Figura 20. Seja que hipoteticamente 4 regiões foram encontradas pelo algoritmo RPN, conforme a Figura 21:

Figura 21 – Imagem com os *bounding boxes* candidatos ou regiões preditas.



Autoria (ANALYTICSVIDHYA.COM, 2020)

As *IoU* dos *bounding boxes* 1 e 2 são menores que 0,5, enquanto as IoU dos *bounding boxes* 3 e 4 são aproximadamente maiores que 0,5. Assim, os *bounding boxes* 3 e 4 são regiões de interesse, enquanto os *bounding boxes* 1 e 2 serão negligenciados.

4. Para cada IoU que contém um objeto é retornada uma máscara de segmentação candidata. A máscara tem tamanho 28×28 que é então ampliada posteriormente. Por fim, as máscaras candidatas passam por uma camada totalmente conectada para classificação e geram as máscaras finais para os objetos.

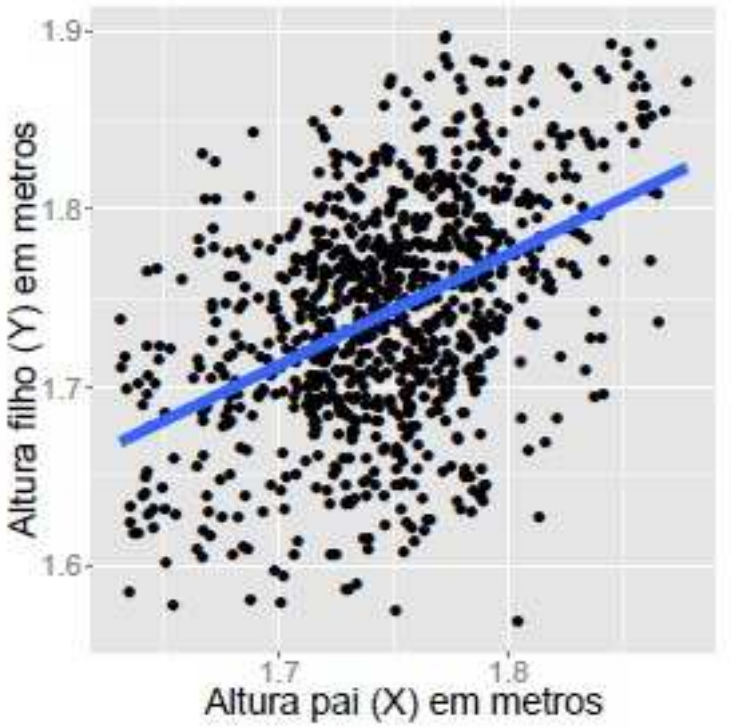
3.5 Algoritmos não supervisionados

A maioria dos problemas de aprendizado de máquina se enquadra em uma das duas categorias: fazer predições (aprendizado supervisionado) ou aprender estruturas (aprendizado não supervisionado). No domínio do aprendizado supervisionado, para cada observação do conjunto de dados $\{x_i, i = 1, ..., n\}$, existe uma resposta associada y_i, onde n é o total de observações. Dessa forma, é desejável encontrar um modelo que relacione a resposta as variáveis independentes, com o objetivo de prever com precisão a resposta para futuras observações (previsão) ou melhor entender a relação entre a resposta e as variáveis independentes (inferência) (BISHOP, 2006).

Seja o seguinte exemplo: deseja-se prever a altura de um filho y_i com base na altura de seu pai x_i, e foram coletadas amostras $\{x_i, i = 1, ..., n\}$ com observações sobre n indivíduos. Com base nesses dados, é modelada uma função de predição $g(x)$: dado que a altura do pai de um indivíduo é x_i, $g(x)$ representa a predição sobre a altura do filho y_i. Suponha predizer a altura de um filho cujo pai tem $x = 1,80m$. A predição neste caso é de $g(1,80) = 1,77$. Se foi observada uma altura de $1,76m$, como quantificar o erro? Uma maneira de se fazer isso é usando o erro quadrático, portanto o erro incorrido por g para prever $x = 1,80$ é de $(g(x) - y)^2 = (1,76 - 1,77)^2 = 0,0001$. Portanto, quanto maior o valor de $(g(x) - y)^2$, pior é a predição. Desta maneira é possível quantificar quão boa g é apenas pelo par (x, y).

A partir desse exemplo é possível destacar os principais objetivos dos métodos de aprendizado supervisionado: (i) construir g pelo par (x, y) de modo a se obter boas predições, e (ii) saber quantificar o quão boa uma (função de) predição é. Uma maneira simples de se criar uma função de predição g neste caso é através de uma regressão linear, como ilustrado na Figura 22.

Figura 22 – Exemplo de aprendizado supervisionado.

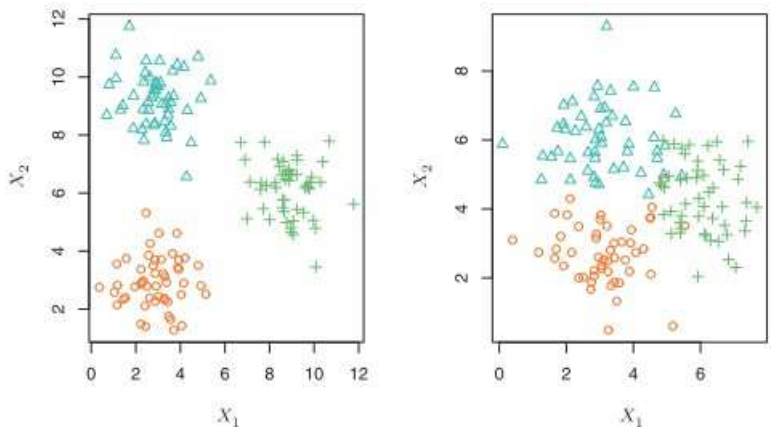


Autoria (JAMES et al., 2014)

Em contraste, o aprendizado não supervisionado descreve uma situação um pouco mais desafiadora em que para cada observação $i = 1, ..., n$, é observado um vetor x_i, mas nenhuma resposta associada y_i. Não é possível ajustar um modelo para uma saída pré-definida, uma vez que não existe uma variável de resposta a ser prevista. A situação é chamada de variável de resposta não supervisionada, porque não é capaz de supervisionar a análise (JAMES et al., 2014).

As ferramentas do aprendizado não supervisionado são: o ranqueamento e o agrupamento. O objetivo do agrupamento é determinar, com base em $x_1, ..., x_n$, se as observações se enquadram em grupos relativamente distintos, conforme ilustrado na Figura 23 (JAMES et al., 2014).

Figura 23 – Um conjunto de dados envolvendo três *clusters*. Cada *cluster* é mostrado usando um símbolo colorido diferente. Esquerda: Os três *clusters* estão bem separados. Nesse cenário, uma abordagem de *cluster* deve identificar com êxito os três grupos. Direita: Há uma sobreposição entre os grupos. A tarefa de identificar os *clusters* é mais desafiadora.



Autoria (JAMES et al., 2014)

Conforme informado por Pimentel et al. (2014), a detecção de novidade é um problema não supervisionado, pois não é possível treinar o modelo com todos os possíveis exemplos de novidades. Isso acontece pela rara ocorrência das novidades, uma vez que para ser considerado novo não pode haver ocorrências de acordo com a necessidade temporal da aplicação. No entanto, ainda é importante treinar o algoritmo não supervisionado, ou para que se faça ajustes de parâmetros ou maximização de métricas de desempenho a fim de evitar *overfitting* ou *underfitting* (JAMES et al., 2014).

Há muitos compromissos associados com a escolha do modelo, por exemplo, modelos altamente complexos (com muitos parâmetros) podem sofrer de *overfitting*. Já modelos simples podem ser construídos a partir de uma boa intuição e entendimento dos dados. Por outro lado, modelos simples se ajustam pobremente aos dados (JAMES et al., 2014).

Claramente, a escolha do modelo de dados é crucial para detecção das novidades. Uma escolha incorreta do modelo de dados pode levar a resultados insuficientes. Por exemplo, modelos totalmente gerativos, como modelos de misturas de gaussianas, podem não funcionar bem, se os dados não se ajustam as hipóteses gerativas do modelo, ou se um número suficiente de exemplos não está disponível para aprender os parâmetros do modelo (AGGARWAL, 2016). Desta forma, foram selecionados diferentes algoritmos não supervisionados do estado da arte e *baseline* para comparar com o desempenho do método proposto, como visto na Tabela 2. Esses algoritmos foram selecionados por produzirem os melhores resultados de cada abordagem.

Tabela 2 – Algoritmos de aprendizado de máquina do estado da arte para detecção de novidades.

Tipo de Abordagem	Algoritmo de aprendizado
Proximidade	kNN (ANGIULLI; PIZZUTI, 2002)
	HBOS (GOLDSTEIN; DENGEL, 2012)
Combinação	Feature Bagging (LAZAREVIC; KUMAR, 2005)
Conexionista	Autoencoder (CHEN et al., 2017)
Subespaços	Isolation Foresting (LIU; TING; ZHOU, 2012)

3.6 kNN

O kNN (*k-Nearest Neighbor*) é um algoritmo de aprendizado supervisionado que possui uma abordagem baseada em proximidade para detectar as novidades. De forma geral, o kNN é um algoritmo não paramétrico em que pega os k vizinhos mais próximos e cada vizinho constitui um voto para seu rótulo. Ganha aquele que possuir mais votos.

O kNN adotado neste trabalho foi adaptado para uma versão não supervisionada. A ideia da abordagem é modelar as novidades como dados isolados dos demais dados, baseado em uma função da distância ou similaridade (AGGARWAL, 2016). A Equação 3.7 descreve a estratégia do kNN não supervisionado:

$$s(x,k) = max(\sum_{i=1}^{k} dist(x_i, x_i')) \tag{3.7}$$

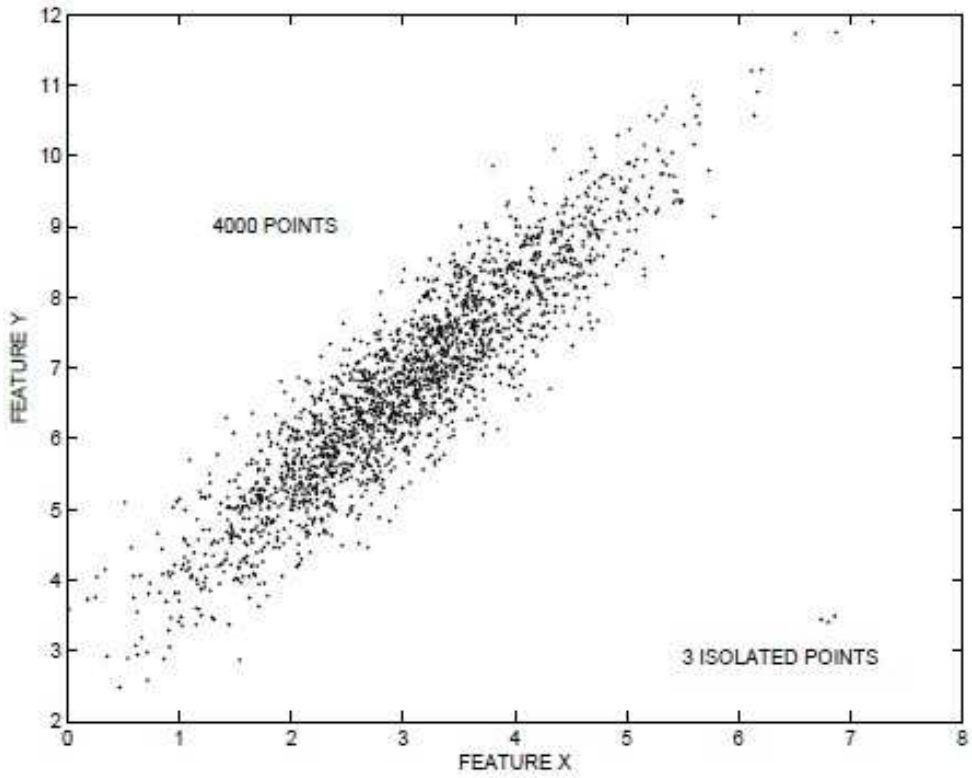
A Equação 3.7 gera uma pontuação s que é baseada no cálculo da distância euclidiana entre uma instância x e sua vizinhança k. A distância para os k vizinhos mais próximos é reportada como pontuação para seleção de novidades ou *outliers*. Quanto maior a distância, maior a chance de ser considerada uma novidade.

Considere o caso ilustrado da Figura 24, que contém um grande grupo contendo 4000 dados e um pequeno conjunto de dados isolados, além de três dados que estão espaçados e relacionados a novidades ou *outliers*. Essas situações são bastante comuns, porque os *outliers* causados pelo mesmo (raro) processo podem resultar em pequenos conjuntos de pontos que são quase idênticos. Neste caso, pontos dentro de um conjunto anômalo são próximos um dos outros, e podem não ser distinguidos em distâncias 1-vizinho. Essas anomalias são frequentemente difíceis de distinguir dos ruídos, usando certos tipos de algoritmos que não são sensíveis ao comportamento global dos dados. Por outro lado, a abordagem dos k-vizinhos pode algumas vezes ser efetiva. No caso da Figura 24, os conjuntos de anomalias relacionadas podem ser identificadas usando $k \geq 3$.

A pontuação dos k vizinhos mais próximos fornece um *score* dos dados, e pode ser decisivo em causar *overfitting* e *underfitting* no modelo. Além disso, o método pode ser computacionalmente caro, porque é necessário determinar os k vizinhos mais próximos

de cada dado do conjunto de dados, que requer $O(N^2)$ operações para um conjunto de amostras contendo N amostras (AGGARWAL, 2016).

Figura 24 – Pequenos grupos de anomalias que podem ser um desafio para a correta identificação.



Autoria (AGGARWAL, 2016)

3.7 HBOS

Assim como o kNN, o HBOS (*Histogram-Based Outlier Scoring*) é um método de aprendizado de máquina baseado em proximidade. Os métodos baseados em proximidade podem ser aplicados de três formas: baseados em agrupamento (*clustering*), densidade e no vizinho mais próximo. O HBOS (GOLDSTEIN; DENGEL, 2012) é um algoritmo que utiliza a técnica de densidade. Os algoritmos baseados em densidade segmentam o espaço de amostras em histogramas. Na maioria dos métodos propostos, uma largura fixa do histograma é determinada ou as larguras são definidas usando parâmetros conhecidos como *bin*.

Se os dados forem multivariados, um histograma para cada característica deve ser calculado, individualmente pontuado e combinado ao final. O HBOS é muito eficiente no processamento de grande conjunto de dados, embora sua combinação de métodos univariados não seja capaz de modelar as dependência entre as características.

O primeiro passo do algoritmo é construir um histograma univariado para cada característica. Se a característica é um dado categórico, uma contagem simples dos valores de cada categoria é realizada e a frequência relativa (altura do histograma) é calculada. Para características numéricas, dois métodos diferentes podem ser usados em relação a largura dos *bins*: (1) Histogramas estáticos ou (2) Histogramas dinâmicos.

A técnica estática usa n *bins* de largura igual ao longo do intervalo de valores. A frequência de amostras de cada *bin* é usada como uma estimativa de densidade (altura dos *bins*). A largura dinâmica é determinada com a seguinte estratégia: valores são ordenados e então uma quantidade fixa de valores são agrupados em um único *bin*, a largura de cada *bin* será definido por $\frac{N}{n}$, onde N é a quantidade de instâncias ou observações totais e n o número de *bins*. Como a área de um *bin* em um histograma representa o número de observações, então é a mesma para todos os *bins*. A largura de um *bin* é definida pelo primeiro e último valor e a altura de cada *bin* individual pode ser calculada. Isto significa que os *bins* que cobrem um intervalo grande de faixa de valores tem alturas menores e representam uma densidade menor. Entretanto, há uma exceção, sob certas circunstâncias, mais de n instâncias de dados podem ter exatamente o mesmo valor, por exemplo, se a característica é um inteiro e uma distribuição de cauda longa foi estimada.

A razão pela qual ambos os métodos (dinâmicos e estáticos) estão disponíveis é devido ao fato de existirem distribuições de dados muito diferentes no mundo real. Especialmente quando há grandes intervalos entre valores (intervalos sem instâncias). Uma vez que a tarefa de detecção de anomalias geralmente envolve essas lacunas em faixas de valores devido ao fato que *outliers* estão distantes dos dados normais, é recomendado o uso do modo dinâmico (GOLDSTEIN; DENGEL, 2012), especialmente se a distribuição é desconhecida ou de cauda longa. Além disso, também o valor de n precisa ser configurado. Uma regra frequentemente usada é configurar n como a raiz quadrada do número de instâncias.

Para cada dimensão d, um histograma individual é calculado (independente se é categórico, tamanho fixo ou dinâmico), onde a altura de cada *bin* representa uma estimação da densidade. Os histogramas são normalizados de tal forma que a altura máxima seja 1. Isso assegura um peso igual de cada característica ao *outlier score*. Finalmente, o *score* de cada instância x é calculado usando a altura correspondente dos *bins*, onde a instância está localizada, conforme Equação 3.8.

$$s(p) = \sum_{i=1}^{d} \log \frac{1}{hist_i(x)} \tag{3.8}$$

A pontuação *score* é a multiplicação do inverso das densidades estimadas. A Equação 3.8 assumi independência das características, dessa forma se comporta similar ao encontrado no modelo de probabilidade discreta Naive Bayes. É importante destacar

que ao invés da multiplicação, é realizada a soma dos logaritmos que é basicamente o mesmo que ($\log(a.b) = \log(a) + \log(b)$). Essa operação não muda a ordem da pontuação. Esse recurso é aplicado devido a ser menos sensível a ruídos, pois a precisão de ponto flutuante em distribuições extremamente desbalanceadas causam pontuações muito altas (GOLDSTEIN; DENGEL, 2012).

3.8 *Feature Bagging*

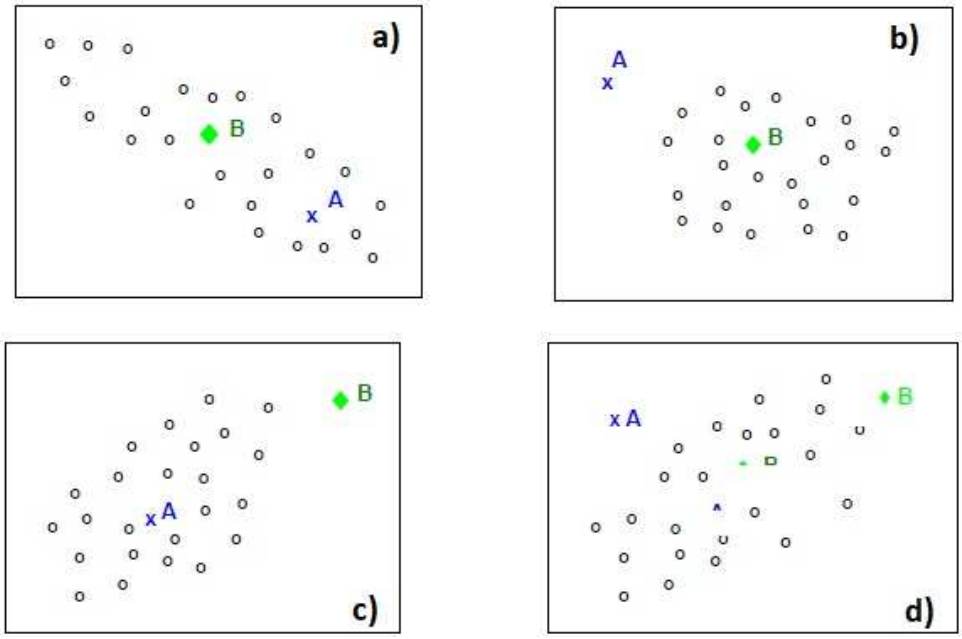
Muitos algoritmos de detecção de *outlier* (AGGARWAL; YU, 2001), (BREUNIG et al., 2000), (KNORR; NG, 1998) e (YU; SHEIKHOLESLAMI; ZHANG, 2002) tentam detectar *outlier* calculando distância em espaços dimensionais completos. Entretanto, em espaços dimensionais grandes, as amostras são muito esparsas e o conceito de similaridade pode não ser significativo. De fato, devido à natureza esparsa das distribuições em um espaço de alta dimensão, a distância entre amostras pode ser bastante similar. Assim, usando a noção de similaridade em espaços de alta dimensão, cada amostra pode ser considerado um potencial *outlier* (BEYER et al., 1999) (AGGARWAL; YU, 2001).

Além disso, para muitos algoritmos apenas um subconjunto de atributos é útil para detectar comportamentos anômalos. No exemplo mostrado na Figura 25, os registros A e B podem ser vistos como *outliers* apenas quando duas dimensões são selecionadas, enquanto nas outras duas projeções mostram um comportamento médio. Além disso, quando um número significativo de características na base de dados são consideradas ruidosas, encontrar *outliers* em todas as dimensões tipicamente não resultam em uma detecção efetiva, ao mesmo tempo, é difícil identificar características relevantes onde os *outliers* podem ser observados.

Outro ponto conhecido em aprendizado de máquina é que *ensembles* de classificadores podem ser efetivos na melhora geral da predição. Essas técnicas de combinação tipicamente manipulam dados de treinamento usando classificadores únicos (ex: *bagging* (BREIMAN, 1996b), *boosting* (FREUND; SCHAPIRE, 1996)) ou rótulos de classes (ex: ECOC (*Error-correcting output coding*) (KONG; DIETTERICH, 1995b)). Em geral, um *ensemble* de classificadores deve ser diverso e preciso a fim de melhorar a predição geral. A diversidade assegura que todos os classificadores não cometam os mesmos erros (TUMER; GHOSH, 1996) (NADA, a).

É importante notar que o problema de combinação de algoritmos de detecção de *outliers* não é exatamente o mesmo problema de *ensemble* de classificadores devido a várias razões. Primeiro, em *ensemble* de classificadores, os algoritmos de classificação lidam com combinação de saídas discretas (rótulos das classes) geralmente usando tipos diferentes de técnicas de votação. Na combinação de algoritmos de detecção de *outliers*, as pontuações ou *rankings* de algoritmos são combinados ao invés de rótulos das classes,

Figura 25 – Diferentes projeções em duas dimensões do espaço podem ou não revelar diferentes conjuntos de *outliers*.



Autoria (LAZAREVIC; KUMAR, 2005)

embora alguns *ensembles* também combinem *rankings* (ou probabilidade das classes estimadas) de classificadores por meio da média ou outra função matemática. Segundo, classificadores que são combinados geralmente têm conhecimento completo dos dados de treinamento e seus rótulos (aprendizado supervisionado), enquanto algoritmos de detecção de *outliers* normalmente lidam apenas com dados sem qualquer rótulo (aprendizado não supervisionado). Entretanto, alguns *ensembles* que não usam rótulos (exemplo *bagging*) são bastante similares aos algoritmos de detecção de *outliers*. Finalmente, certos *ensembles* (exemplo *boosting*) podem controlar o processo de combinação observando a taxa de erro, que não é possível em combinação de algoritmos de detecção de *outliers* uma vez que o rótulo não é determinado ou não são conhecidos com antecedência quais dados são realmente *outliers*.

Como solução para os problemas relatados anteriormente, *Featuring Bagging* (LAZAREVIC; KUMAR, 2005) é proposto para dados de alta dimensão e ruidosos. Ao contrário da abordagem padrão *bagging*, onde o modelo de classificação/regressão são combinados usando amostras aleatórias, nesta abordagem algoritmos de detecção de *outliers* são combinados e sua diversidade é melhorada pela amostragem aleatória de subconjuntos de características do conjunto de dados original. Os resultados combinados de múltiplos algoritmos de detecção de *outliers* são obtidos usando diferentes conjuntos de características. Cada algoritmo usa um pequeno conjunto de características que são selecionados aleatoriamente a partir do conjunto de características originais. Como resultado, cada detector de *outliers* identifica diferentes *outliers*, e assim atribui para todos os dados

pontuações (*scores*). A pontuação calculada por cada algoritmo individual é combinada a fim de detectar os *outliers* com maior confiabilidade.

Duas técnicas são propostas para combinar algoritmos de detecção de *outliers*. O algoritmo geral é mostrado no Algoritmo 1. O procedimento para combinar técnicas de detecção de *outliers* procede em uma série de j rodadas, embora essas rodadas possam ser executadas em paralelo para uma execução mais rápida. Em cada rodada j, o algoritmo de detecção de *outliers* é chamado e é apresentado a ele um subconjunto diferente de características Q que é usado no cálculo da distância. O conjunto de características Q é aleatoriamente selecionado dos dados originais, de modo que o número de características em Q é também aleatoriamente escolhido entre $\lfloor d/2 \rfloor$ e $(d-1)$, onde d é o número de características dos dados originais. As Q características são aleatoriamente selecionadas sem reposição do conjunto de características originais.

Cada algoritmo de detecção, como resultado, gera vetores de pontuação de diferentes *outlier* s que reflete a probabilidade de cada dado do conjunto A ser um *outlier*. Por exemplo, se $s(i) > s(j)$, o dado x_i tem probabilidade maior de ser *outlier* que o dado x_j. Ao final do procedimento, depois de J rodadas, há J vetores de pontuação de *outliers*, cada um correspondendo a um único algoritmo de detecção de *outlier*. A função *COMBINE* (Algoritmo 1) é então usada para combinar J vetores de pontuação s, $j = 1, 2 \cdots J$, em um vetor de *score* único s_{final}, que é usado para atribuir uma probabilidade final de ser um *outlier* para cada dado do conjunto.

O problema de combinação de pontuação de *outliers* é conceitualmente bastante similar ao problema de motores de busca em que diferentes classificações (*rankings*) retornadas por mecanismos de buscas são combinadas a fim de fornecer as páginas que são mais relevantes para os termos buscados. Em ambos problemas, não há um rótulo que ajude a entender a relevância dos resultados da pesquisa, e o *ranking* dos algoritmos individuais é importante no processo de combinação, uma vez que fornecem a noção da relevância do resultado. Motivado por várias abordagens usadas nos mecanismos de buscas, são exploradas duas variantes da função *COMBINE* que integra as saídas de múltiplos algoritmos de detecção de *outliers*: *Breadth First* e Soma Cumulativa, que são ilustrados nos Algoritmos 2 e 3. O algoritmo *Breadth First* tem por objetivo selecionar o *score* de maior pontuação. O algoritmo Soma Cumulativa retorna o *score* como o somatório de todos os *scores*.

3.9 *Isolation Foresting*

O algoritmo *Isolation Forest* ou *iForest* (LIU; TING; ZHOU, 2012) realiza um processo em dois estágios. No primeiro estágio (treinamento) são construídas árvores isoladas usando amostras do conjunto de treinamento. No segundo estágio (teste) os dados

Algoritmo 1 FBagging(S,J)

Entrada: $S\{(x_1, y_1), \ldots, (x_n, y_n)\}$ $x_i \in \boldsymbol{X^d}$ - dados de entrada
$y_i \in Y = \{C, NC\}$, onde C - *outliers*, NC - classes normais
d - dimensionalidade
Saída: s_{final} - vetor de pontuação final

1: **Inicializar:** Normaliza o conjunto S
2: **Inicializar:** $|Q|$ tem distribuição uniforme entre $\lfloor d/2 \rfloor$ e $(d-1)$.
3: **for** `j=1 to J` **do**
4: Escolha aleatoriamente o tamanho do subconjunto de características $|Q|$.
5: Pegue aleatoriamente sem reposição $|Q|$ e crie o subconjunto de características Q.
6: Aplique o algoritmo de detecção de *outlier* O_t empregando o subconjunto Q.
7: A saída do algoritmo O_t é o vetor de pontuação s
8: Combina s, e cria pontuação final $s_{final} = COMBINE(s)$
9: **end for**
10: **return** s_{final}

Algoritmo 2 BreadthFirst(s)

Entrada: $s\{1,\ldots,T\}$, sendo m o tamanho dos dados do *Dataset* e de cada vetor s
1: **Ordene** todos os *scores* de vetores de s e armazene em s_{final} e retorne o índice Ind_t, de forma que $s_{final}(1)$ tenha o maior *score* e $Ind_{final}(1)$ tenha o índice do registro de s com o maior *score* de s_{final}
2: **for** `i=1 to m` **do**
3: **for** `t=1 to T` **do**
4: Se o índice Ind_t está ranqueado em uma posição i pelo algoritmo t e tem um *score* s(i) que não existe no vetor Ind_{final}.
5: Insere $Ind_t(i)$ no final do vetor Ind_{final}.
6: Insere $s(i)$ no final do vetor s_{final}.
7: **end for**
8: **end for**
9: **return** Ind_{final} e s_{final}

Algoritmo 3 Cumulativa(s)

Entrada: $s\{1,\ldots,T\}$, sendo m o tamanho de cada vetor s
1: **Some** todos os *scores* de s armazene em s_{final}
2: **for** `i=1 to m` **do**
3: $s_{final}(i) = \sum_{t=1}^{T} s(i)$.
4: **end for**
5: **return** s_{final}

de teste passam através das árvores isoladas para obter uma pontuação de anomalia para cada dado.

No estágio de treinamento, árvores ($iTrees$) são construídas recursivamente particionando o conjunto de treinamento até que dados estejam isolados ou uma altura específica da árvore seja atingida, resultando em um modelo parcial. As ($iTrees$) têm uma estrutura equivalente a uma árvore binária de busca. A altura limite da árvore l é configurada pelo tamanho da amostra ψ : $l = ceiling(\log_2 \psi)$, que é a altura média da árvore. A lógica

do crescimento da árvore denota interesse em dados que tenham uma média menor que o comprimento do caminho médio, esses pontos são mais prováveis de serem anomalias. Detalhes do estágio de treinamento estão nos Algoritmos 4 e 5.

Há dois parâmetros de entrada no algoritmo *iForest*, que são o tamanho da amostragem ψ e o número de árvores t. A amostragem ψ controla o tamanho dos dados de treinamento. Quando ψ aumenta para um valor desejado, *iForest* detecta *outliers* de forma confiável e não há necessidade de aumentar ainda mais, pois aumenta o tempo de processamento e o tamanho de memória sem ganho no desempenho. Empiricamente, ao configurar ψ para 2^8 ou 256 (LIU; TING; ZHOU, 2012) geralmente é suficiente para realizar detecção de anomalias através de uma grande variedade de dados. O número de árvores t controla o tamanho do *ensemble*. A complexidade de tempo do *iForest* é $O(t(n+\psi)\psi)$ e a complexidade de espaço é $O(t\psi)$, em que $n = |X|$ é a quantidade total dos dados de teste.

Algoritmo 4 iForest(X,t,ψ)

Entrada: X - dados de entrada, t - núm. de árvores, ψ - tamanho da amostragem
Saída: Conjunto $iTrees$

Inicializar: Conjunto $iTrees$
set altura limite l= $ceiling(\log_2 \psi)$
for `i=1 to t` **do**
    $X' \leftarrow sample(X, \psi)$
    $Forest \leftarrow Forest \cup iTree(X', 0, l)$
end for
return $Forest$

Algoritmo 5 iTree(X,e,l)

Entrada: X - conjunto de dados de entrada, e - altura atual da árvore, l - altura limite
Saída: uma $iTree$

if $e \geq$ l ou $|X| \leq 1$ **then**
    return $exNode$ { $Size \leftarrow |X|$ }
else
    Seja Q a lista de atributos em X
    Aleatoriamente selecione atributos em $q \in Q$
    Aleatoriamente selecione dados p de max a min dos atributos q de X
    $X_l \leftarrow filter(X, q < p)$
    $X_r \leftarrow filter(X, q \geq p)$
    return $inNode$ { $Left \leftarrow iTree(X_l, e+1, l)$,
        $Right \leftarrow iTree(X_r, e+1, l)$,
        $SplitAtt \leftarrow q$,
        $SplitValue \leftarrow p$ }
end if

Sendo X o conjunto de dados de entrada e x uma instância desse conjunto, a pontuação de anomalia s é decorrente de $E(h(x))$, que é a média de $h(x)$ da coleção de árvores isoladas ($iForest$). $E(.)$ é a função de valor esperado ou média. $h(x)$ retorna o

comprimento do caminho para cada instância x. O comprimento é medido pelo número de arestas de x que atravessa uma $iTree$ de um nó raiz até que o percurso seja terminado em um nó externo (nó sem filhos). Curtos e longos caminhos exprimem alto grau de isolamento de um nó e consequentemente indicam uma anomalia.

Usando a função $PathLegth$, um único caminho de $h(x)$ é calculado pela contagem do número de arestas e, do nó raiz até o nó terminal, conforme a instância x passa pela $iTree$. Quando o percurso atinge uma altura limite $hlim$, o valor de retorno é e mais um fator de ajuste $c(\psi)$. O ajuste é responsável pela estimativa de um comprimento médio de uma subárvore aleatória que poderia ser construída além do limite de altura da árvore. Após $h(x)$ ser obtida para cada árvore do *ensemble*, então é calculada a esperança E de $h(x)$, e por final uma pontuação de anomalia é produzida pelo cálculo $s(x, \psi)$ nas Equações 3.9 e 3.10. O detalhamento da função $PathLegth$ pode ser encontrado no Algoritmo 6.

$$s(x, \psi) = 2^{-\frac{E(h(x))}{c(\psi)}} \tag{3.9}$$

O valor de $c(\psi)$ é um fator de normalização baseado na média da altura da árvore.

$$c(\psi) = \begin{cases} 2H(\phi - 1) - (2(\psi - 1)/n) & \text{for } \psi > 2 \\ 1 & \text{for } \psi = 4 \\ 0 & \text{caso contrário} \end{cases} \tag{3.10}$$

$H(x)$ é uma função que produz um número harmônico e pode ser estimado por $ln(x) + 0,5772156649$ (constante de Euler).

Algoritmo 6 PathLegth(x,T,hlim,e)

Entrada: x - uma instância, T - uma $iTree$, $hlim$- altura limite, e - comprimento caminho
Saída: comprimento do caminho de x

1: **if** T é um nó externo ou $e \geq hlim$ **then**
2: **return** $e + c(T.size)$
3: **end if**
4: $a \leftarrow T.splitAtt$
5: **if** $x_a < T.splitValue$ **then**
6: **return** $PathLength(x, T.left, hlim, e + 1)$
7: **else** $x_a \geq T.splitValue$
8: **return** $PathLength(x, T.right, hlim, e + 1)$
9: **end if**

3.10 *Autoencoder*

Um *autoencoder* é um tipo especial de rede neural multicamadas que realiza uma redução de dimensionalidade não linear e hierárquica dos dados (GOODFELLOW;

YOSHUA; COURVILLE, 2016) (CHEN et al., 2017). Normalmente, o número de nós na camada de saída é o mesmo que da camada de entrada, e a arquitetura é em camadas e simétrica. O objetivo de um *autoencoder* é treinar a saída para reconstruir a entrada tão próxima quanto possível. Os nós das camadas do meio são em menores números e, portanto, a única forma para reconstruir a entrada é aprender os pesos a fim de que as saídas dos nós nas camadas intermediárias sejam representações reduzidas (CHEN et al., 2017).

Uma vez que o *autoencoder* cria uma representação de redução da dimensão dos dados, então é estabelecida uma abordagem natural para descoberta de *outliers*. A ideia básica é que *outliers* são muito difíceis de serem representados com precisão igual ou maior do que os pontos normais. Portanto, a reconstrução de um *outlier* resulta em um erro grande. Isso fornece uma forma natural de atribuir uma pontuação (*score*) a um dado (AN; CHO, 2015).

Um *autoencoder* é composto de duas partes, como ilustra a Figura 26: uma codificadora e uma decodificadora. Uma rede neural com uma única camada oculta tem uma camada codificadora e uma decodificadora como nas Equações 3.11 e 3.12, onde W e b são os pesos e *bias* da rede neural, respectivamente, e α é uma função de transformação não linear (AN; CHO, 2015).

$$h = \alpha(W_{xh}x + b_{xh}) \tag{3.11}$$

$$z = \alpha(W_{hx}B + b_{hx}) \tag{3.12}$$

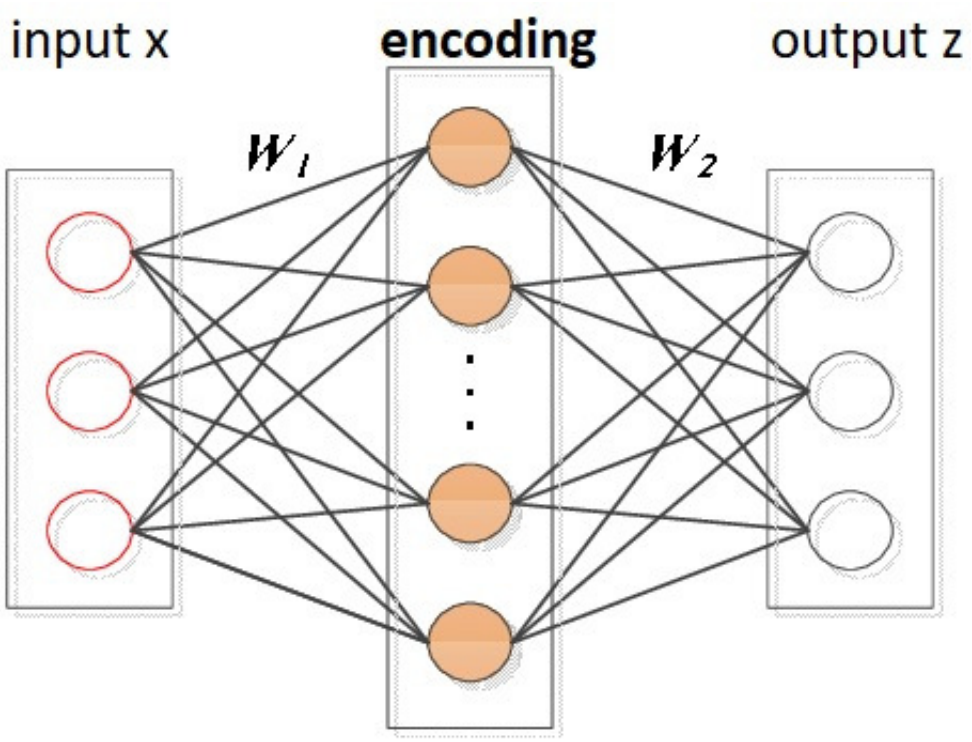
$$e = \|x - z\| \tag{3.13}$$

O codificador na Equação 3.11 mapeia um vetor de entrada x para uma representação oculta B após uma não-linearidade α. O decodificador na Equação 3.12 mapeia a representação oculta B de volta no espaço de entrada original como uma reconstrução pela mesma transformação do codificador. A diferença entre o vetor de entrada original x e a reconstrução z é chamada de erro de reconstrução e, como na Equação 3.13. Um *autoencoder* aprende a minimizar o erro de reconstrução (AN; CHO, 2015).

Em um nível básico, cada nó na rede neural aplica uma função de ativação sobre uma soma ponderada das entradas. Podem ser usadas como funções de ativação a Sigmóide e a ReLU. Especificamente, na primeira camada oculta e na camada de saída, a função Sigmóide é usada. As funções Sigmóide e a ReLU são definidas em 3.14 e 3.15:

$$\alpha(x) = \frac{1}{1 + e^{-x}} \tag{3.14}$$

Figura 26 – Esboço do *autoencoder* totalmente conectado usado na proposta deste trabalho.



Autoria (CHEN et al., 2017)

$$f(x) = max(0, x) \tag{3.15}$$

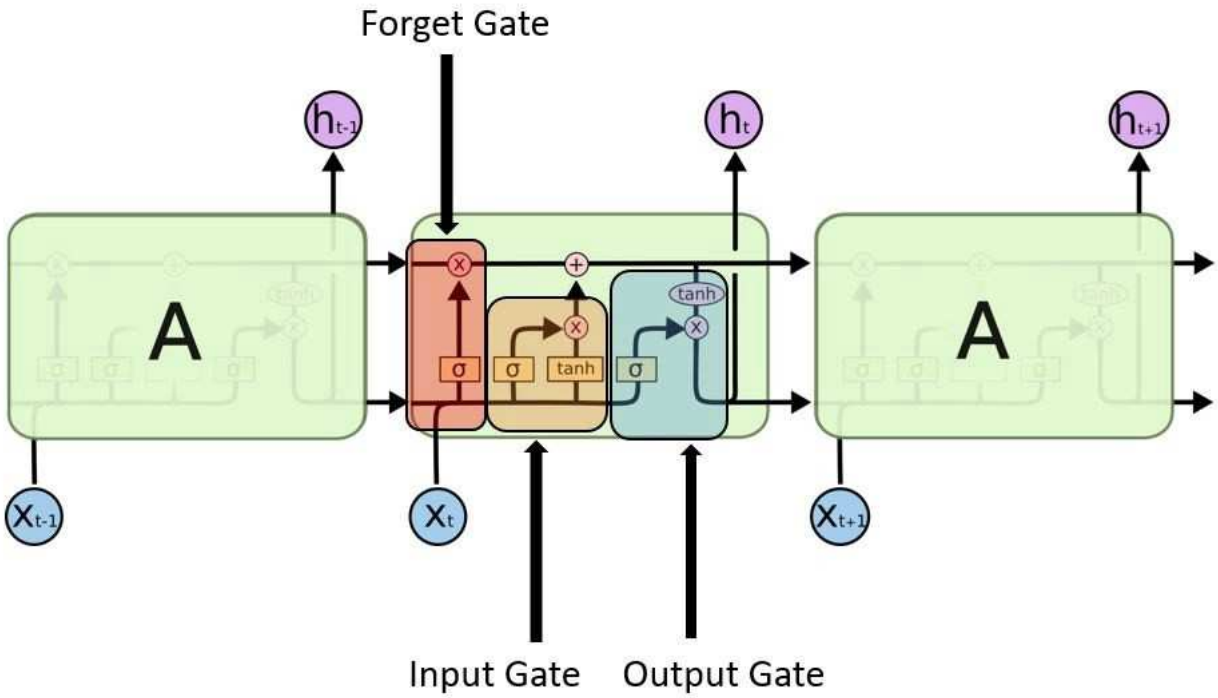
Para todas as outras camadas foi usada, neste trabalho, a função ReLU. Usar funções de ativações diferentes em camadas diferentes visa equilibrar as vantagens e desvantagens das duas funções de ativações. A função Sigmóide é conhecida por causa do problema do desaparecimento do gradiente, e sua operação é muito cara comparada a ReLU. A ReLU é simples e computacionalmente mais eficiente, e não sofre do problema do desaparecimento do gradiente. Entretanto, sofre do problema do *Dying* ReLU, recentemente descoberto por pesquisadores (KRIZHEVSKY; SUTSKEVER; HINTON, 2012b). Isso significa que durante o treinamento, uma atualização de peso desencadeada por um gradiente grande, devido a uma função ReLU, poderia tornar o neurônio (nó) inativo o tempo todo. Finalmente, isso fará com que muitos dos neurônios fiquem em um estado de morte em que nenhum peso seja atualizado e a rede continue a fornecer a mesma saída sobre as iterações. Isso motiva o uso da função Sigmóide nas duas extremidades da rede e a ReLU em outras camadas. Por um lado, o uso das funções Sigmóide assegurará que, mesmo que todo os neurônios da camada do meio estejam inativos, pelo menos haverá duas camadas funcionando corretamente nas duas extremidades da rede, o que garante um melhor cenário (CHEN et al., 2017).

3.11 *LSTM*

LSTM (*Long Short-Term Memory*) é uma arquitetura de rede neural recorrente (RNN) adequada para classificar, processar e prever séries temporais com intervalos de tempo de duração desconhecida. A insensibilidade relativa ao comprimento do intervalo dá uma vantagem à LSTM em relação a outras arquiteturas de RNNs tradicionais (também chamadas *vanilla*), Modelos Ocultos de Markov (MOM) e outros métodos de aprendizado de sequências (deeplearningbook, 2019) (GOODFELLOW; YOSHUA; COURVILLE, 2016).

A LSTM possui uma estrutura em cadeia que contém quatro camadas de interação e diferentes blocos de memória chamados células. Em uma rede LSTM, três portões estão presentes, conforme apresentado na Figura 27. A informação é retida pelas células e as manipulações de memória são feitas pelos portões (*gates*) (deeplearningbook, 2019) (GOODFELLOW; YOSHUA; COURVILLE, 2016).

Figura 27 – Esboço da arquitetura da *LSTM*.



Autoria (Mittal, A., 2019)

A LSTM possui os seguintes *gates* (portões):

- *Forget Gate:* As informações que não são mais úteis no estado da célula são removidas

com o *forget gate*. Duas entradas: x_t (entrada no momento específico) e h_{t-1} (saída da célula anterior ou estado anterior) alimentam o *gate* e são multiplicadas pela matriz de peso W_f, seguida pela adição do *bias* b_f, conforme Equação 3.16. O resultante é passado para uma função de ativação α, chamada de Sigmóide, que fornece um valor real entre 0 e 1. Se para um determinado estado da célula a saída for 0, a informação é esquecida e para a saída 1, a informação é retida para uso futuro (Mittal, A., 2019) (Olah, C., 2015).

$$f_t = \alpha(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.16}$$

- *Input Gate:* A adição de informações úteis ao estado da célula é feita pelo *input gate*, conforme apresentada na Equação 3.17 e pode ser dividida em dois passos. O primeiro passo é decidir quais informações úteis devem ser armazenadas. Para isso, a função α atua filtrando os valores, e portanto decide quais valores serão lembrados, agindo de forma similar ao *forget gate*. A função α usa o estado anterior h_{t-1} e a entrada x_t, que é multiplicada pela matriz de pesos W_i, seguida pela adição do *bias* b_i. O vetor i_t contém as informações úteis ao estado da célula. O segundo passo é criar um vetor com novos valores candidatos que podem ser o estado da memória atual no tempo t. A seguir, a função tangente hiperbólica *tanh* atribui pesos aos valores de entrada, decidindo seu nível de importância variando entre -1 a $+1$. A função *tanh* usa o estado anterior h_{t-1} e a entrada x_t, que é multiplicada pela matriz de pesos W_C, seguida pela adição do *bias* b_C. Os vetores i_t e $\widetilde{C_t}$ são combinados para atualização de um novo estado (Mittal, A., 2019) (Olah, C., 2015).

$$\begin{aligned} i_t &= \alpha(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \widetilde{C_t} &= tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{3.17}$$

- *Output Gate:* A tarefa de extrair informações úteis do estado da célula atual C_t para serem apresentadas como uma saída é feita pelo *output gate*, conforme Equação 3.18. A função α filtra os valores a serem lembrados usando as entradas h_{t-1} e x_t, que é multiplicada pela matriz de pesos W_o, seguida pela adição do *bias* b_o. A função *tanh* atribui pesos aos valores passados, decidindo seu nível de importância, variando entre -1 a 1 e multiplicando pela saída da *Sigmóide*. O vetor o_t é combinado ao estado da memória $\widetilde{C_t}$ no tempo t para apresentar a saída h_t (Mittal, A., 2019) (Olah, C., 2015).

$$\begin{aligned} o_t &= \alpha(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \cdot tanh(C_t) \end{aligned} \tag{3.18}$$

O modelo LSTM usado neste trabalho foi o proposto por Ranjan et al. (2018). O objetivo principal desse modelo é detectar eventos raros usando LSTM Autoencoder. A rede de Ranjan et al. (2018) possui quatro camadas ocultas, todas usando função de ativação ReLU, incluindo a regularização L1 e a heurística Adam como otimizador. Esse código está disponível para livre *download* [1]. O LSTM pode ser usado para detectar *outliers* de maneira semelhante ao *autoencoder* descrito na Seção 3.10, conforme apresentado na Equação 3.19. Os dados são mapeados usando o modelo treinado, e o erro de reconstrução entre o valor real observado e o valor alvo dá origem a um *score* que, dependendo do *threshold*, pode ser considerado novidade. O *threshold* é obtido a partir da média dos 1% dos dados com maior *score*. A diferença entre LSTM *autoencoder* e o *autoencoder* é que na rede LSTM o controle da temporalidade é feito pelos *gates* da arquitetura.

$$s(x) = \sum_{k=1}^{d} ([x_{ij}]_k - [o_{ij}]_k)^2 \,. \quad k \in 1 \ldots m, j \in \ldots n \tag{3.19}$$

Sendo a variável m o número de componentes do *ensemble*. Neste trabalho não é usado *ensemble* de *autoencoders*, então $m = 1$. A variável n é a quantidade total de amostras e d a quantidade total de dimensões. Assim, é denotado o i-th componente do *ensemble* da j-th amostra de entrada como $x_{ij} \in \mathbb{R}$ e a saída reconstruída pelo *autoencoder* como $o_{ij} \in \mathbb{R}$.

3.12 *Word embedding* ou vetores densos

Para que o computador possa realizar tarefas de processamento de linguagem natural é necessário transformar o texto em representações vetoriais, sendo as técnicas mais comuns de transformações: *one-hot*, distributiva, *word embedding*, entre outras. Independente da técnica escolhida, o primeiro passo é construir o dicionário de palavras. O dicionário é um conjunto indexado de todas as distintas palavras do corpus textual em questão.

Antes de construir o dicionário, comumente é realizado um pré-processamento no texto. Uma das formas de pré-processar é lematizar as palavras do texto, que consiste em converter a palavra a sua forma base, por exemplo: andar, andando e andei, o lema do verbo é andar. A Tabela 3 exemplifica um dicionário simples construído.

A forma mais simples de representar uma palavra é usar um vetor *one-hot*. O vetor possui a quantidade de dimensões do tamanho do dicionário. Dessa forma, cada dimensão é representada por uma palavra do dicionário, tendo o valor 1 para a dimensão que corresponde ao *ID* da palavra no dicionário e 0 para o resto (KOEHN, 2010). A seguir, é exemplificada a codificação de palavras usando *one-hot*:

[1] https://github.com/cran2367/autoencoderclassifier/blob/master/autoencoderclassifier.ipynb

Tabela 3 – Exemplo de dicionário.

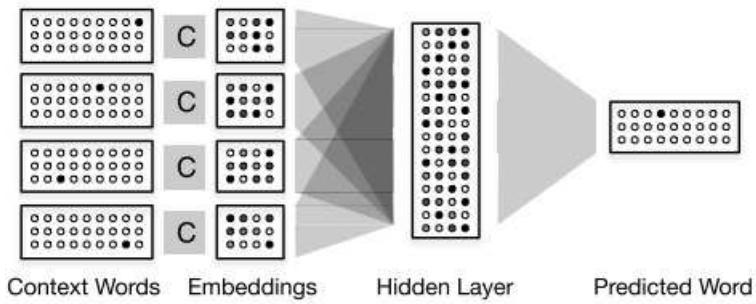
ID	Palavras
0	cachorro
1	escola
2	andar
3	gato
4	comer

- $cachorro = (1, 0, 0, 0, 0)^T$
- $gato = (0, 0, 0, 1, 0)^T$
- $comer = (0, 0, 0, 0, 1)^T$

Essa abordagem *one-hot* sofre de dois principais problemas: os vetores são esparsos e de alta dimensão, ou seja, cada dimensão representa uma característica. Outro problema é que as características são completamente independentes uma das outras. As características do vetor "computador" não tem qualquer similaridade com o vetor "*laptop*" (GOLDBERG, 2016).

Como solução alternativa para os vetores com alta dimensão, utiliza-se a ideia de mapear cada palavra em um espaço de baixa dimensão, que são vetores de pesos de números reais. O algoritmo de redes neurais são os mais comumente utilizados para gerar os *word embedding* ou vetores densos. A Figura 28 ilustra uma típica arquitetura que gera os vetores densos (KOEHN, 2010).

Figura 28 – Arquitetura de um modelo de linguagem neural.



Autoria (KOEHN, 2010)

As palavras do contexto são representadas como um vetor *one-hot* que são as entradas para a rede neural. As palavras do contexto são palavras que se apresentam próximo a palavra alvo em uma frase, exemplo: *O gato subiu na árvore*. Se a palavra alvo escolhida é *gato*, e se as preposições são removidas, as palavras mais significativas do contexto podem ser: *subiu* e *árvore*.

Hipoteticamente, sejam as seguintes frases:

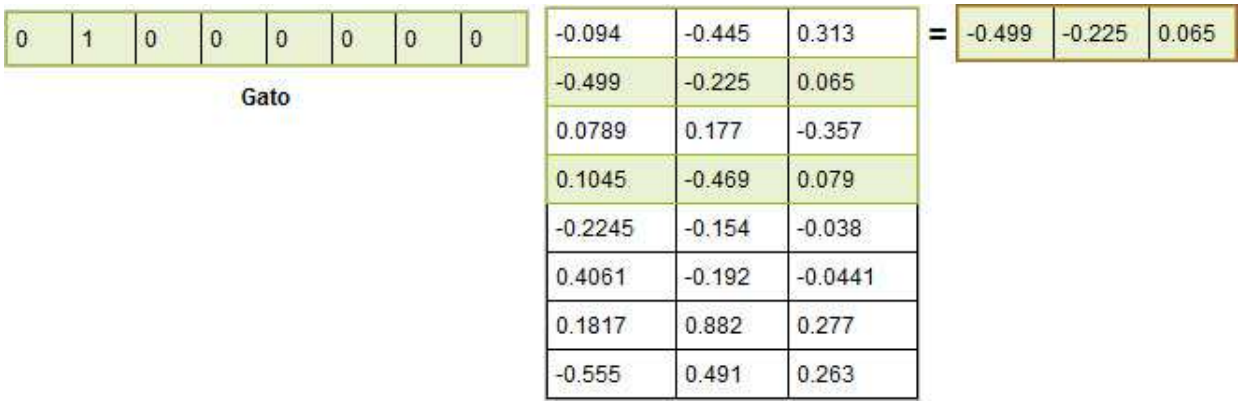
- o cachorro viu o gato
- o cachorro perseguiu o gato
- o **gato subiu** na árvore

O objetivo inicial é fazer com que a rede neural aprenda o relacionamento semântico entre gato e subiu. Portanto, a entrada da rede neural é um vetor *one-hot* da palavra gato e a saída é a probabilidade de subiu estar relacionado a gato. A rede neural será treinada com o objetivo de maximizar a probabilidade das palavras que estão nas proximidades da palavra subiu, por exemplo, gato. Enquanto, palavras distantes ou que nunca aparecem com a palavra subiu terão probabilidade baixa.

O primeiro processamento a ser realizado é a construção do dicionário com o conjunto das diferentes palavras das frases. Dessa forma, para esse exemplo, o dicionário conterá 8 distintas palavras, por conseguinte, se cada palavra for representada como um vetor *one-hot*, então o vetor terá dimensão de tamanho 8. Após a representação das palavras em vetores *one-hot*, ocorre a combinação linear dos pesos da camada oculta da rede neural com a entrada (vetores *one-hot*) (KOEHN, 2010), conforme Equação 3.20 e exemplificado na Figura 29.

$$h^T = x^T * W_{xh} \tag{3.20}$$

Figura 29 – Exemplo da combinação linear da entrada.



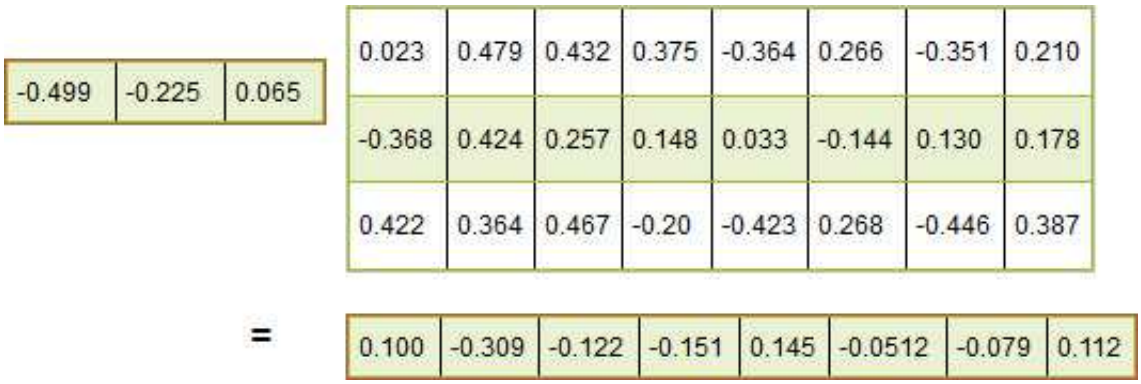
Autoria própria.

Dessa forma, o vetor *one-hot* é projetado em um espaço de baixa dimensão, conforme pode ser visto na combinação linear da entrada com a matriz de pesos da camada oculta.

Próximo passo, é a combinação linear dos pesos com a camada oculta, conforme Equação 3.21 e exemplificado na Figura 30.

$$z = h^T * W_{hx} \tag{3.21}$$

Figura 30 – Exemplo da combinação linear da camada oculta.



Autoria própria.

Por fim, usando função de ativação Softmax (ZHANG; LECUN, 2015) são calculadas as probabilidades para as oito palavras do Corpus. Cada neurônio de saída corresponde a uma palavra. Por exemplo, a probabilidade da saída ser subiu, dado gato como entrada, é de $P(subiu \mid gato) = 0.111$.

$$\delta(z) = y_1 = \frac{e^{Net(z_1)}}{e^{Net(z_1)} + e^{Net(z_2)} + e^{Net(z_3)} + e^{Net(z_4)} + e^{Net(z_5)}} \tag{3.22}$$

Figura 31 – Probabilidade das palavras do Corpus.

Autoria própria.

Toda essa fase da rede neural é conhecida como *feedforward*. A próxima etapa da rede é realizar o *backpropagation*, ou seja, calcular o erro e atualizar as matrizes de pesos. O cálculo do erro é realizado pela minimização da função de perda ou custo (KOEHN, 2010) (MIKOLOV et al., 2013).

Há técnicas diferentes de implementar vetores embutidos usando redes neurais, destas, as mais populares são listadas na Tabela 4.

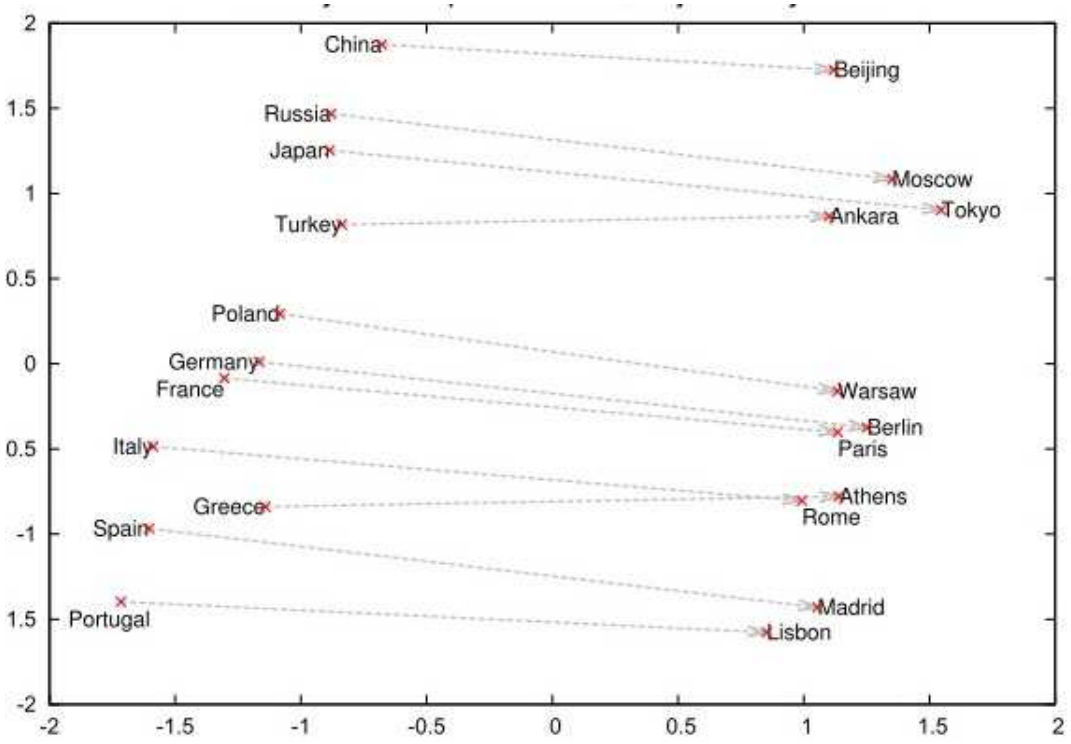
A representação da palavra por meio de vetores densos é interessante porque a rede neural aprende a codificar padrões e regularidades linguísticas. Muitos dos padrões podem

Tabela 4 – Algumas técnicas que geram vetores embutidos.

Técnica	Desenvolvedor	Ano
ELMo (PETERS et al., 2018)	Allen Institute	2018
Bert (DEVLIN et al., 2018)	Google	2018
FastText (JOULIN et al., 2017)	Facebook	2016
Word2Vec (MIKOLOV et al., 2013)	Google	2013
Glove (PENNINGTON; SOCHER; MANNING, 2014)	Stanford	2014

ser representados como uma operação vetorial que resulta em analogias, por exemplo: $\vec{Madrid} - \vec{Spain} + \vec{France} \approx \vec{Paris}$. Observe a Figura 32 em que foram projetados os vetores de cidades e capitais em um gráfico de duas dimensões. Os vetores gerados pela rede neural com 1000 dimensões passaram por uma redução de dimensionalidade usando PCA (*Principal Component Analysis*). A Figura 32 ilustra a habilidade do modelo de automaticamente organizar os conceitos e aprender implicitamente os relacionamentos (MIKOLOV et al., 2013).

Figura 32 – Vetores densos de países e capitais projetados usando a técnica PCA.



Autoria (MIKOLOV et al., 2013)

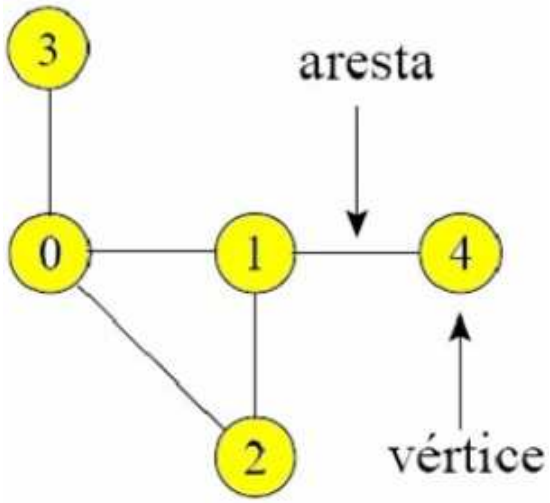
Por enquanto, foi apresentado apenas redes neurais que produzem vetores representantes de palavras. Em processamento de linguagem natural, é necessário realizar operações vetoriais com os vetores das palavras para a representação de uma frase completa, como exemplos: média, concatenação, soma, convolução, entre outras. Alguns autores sugerem hipóteses da melhor representação de frases. De acordo com Le e Mikolov (2014), sim-

plesmente a média de vetores é insuficiente para tarefas de análise de sentimento, porque falha no reconhecimento de fenômenos linguísticos sofisticados, como sarcasmos. Kenter, Borisov e Rijke (2016) relatam que simplesmente calcular a média provou ser uma técnica eficiente para uma grande quantidade de tarefas, como tarefas de similaridade entre textos curtos. Uma variante seria ponderar os vetores de palavras com seus respectivos vetores TF-IDF. Socher et al. (2013) combinaram os vetores de palavras usando análise sintática, o qual funcionou bem para tarefas de análise de sentimento. Nesta pesquisa foi utilizada a técnica de média vetorial para representação dos *tweets*, já que em geral os textos são curtos.

3.13 Grafos

A teoria dos grafos é um ramo da matemática que estuda as relações entre os objetos de um determinado conjunto. Para tal, são empregadas estruturas chamadas de grafos, $G(V, E)$, na qual V é um conjunto não vazio de objetos denominados vértices e E é um subconjunto de pares não ordenados de V, chamados arestas, conforme ilustrado na Figura 33.

Figura 33 – Exemplo de um grafo.



Autoria própria

Dependendo da aplicação, arestas podem ou não possuir direção. Em um grafo, pode ser permitido ou não arestas serem ligadas a mais de um vértice, que, por sua vez, ainda pode possuir um peso (numérico) associado. E caso as arestas possuam uma direção associada (indicada por uma seta na representação gráfica), então há um dígrafo (grafo orientado).

Os grafos são geralmente representados graficamente da seguinte maneira: é desenhado um círculo para cada vértice, e para cada aresta é desenhado um arco conectando

suas extremidades. Entretanto, convenções alternativas para a representação de grafos podem ser adotadas, que incluem representações por adjacência como empacotamento de círculos, onde vértices são representados como regiões disjuntas no plano e arestas são representadas por adjacências entre regiões.

Na área da Inteligência Artificial, muitos algoritmos heurísticos representam o espaço de busca do problema em forma de grafos, com o objetivo de representar seus estados de forma organizada.

A seguir, são apresentados alguns tipos de grafos que podem ser úteis no entendimento da representação da janela temporal de novidades deste trabalho: (GOLDBARG; GOLDBARG, 2012):

- Grafo simples: é um grafo não direcionado, sem laços e com no máximo uma aresta entre quaisquer dois vértices (sem arestas paralelas). Para um grafo simples, o número de vizinhos de um vértice é igual ao seu grau (valência).
- Grafo completo: é um grafo simples em que, para cada vértice do grafo, existe uma aresta conectando este vértice a cada um dos demais, ou seja, todos os vértices do grafo possuem mesmo grau. Um grafo completo de μ vértices é frequentemente denotado por k_μ. Ele tem $\mu(\mu-1)/2$ arestas (correspondendo a todas as possíveis escolhas de pares de vértices).
- Grafo nulo: é o grafo cujo conjunto de vértices é vazio.
- Grafo vazio: é o grafo cujo conjunto de arestas é vazio.
- Grafo trivial: é o grafo que possui apenas um vértice e nenhuma aresta.
- Grafo regular: é um grafo em que todos os vértices possuem o mesmo grau.
- Grafo direcionado ou orientado: um grafo é dito direcionado quando o sentido das ligações entre os vértices é importante. Quando a família de arestas é formada por pares ordenados é dito que o grafo é orientado ou direcionado. Nesse caso, ao desenhar o grafo, deve-se usar uma seta partindo do primeiro elemento do par até o segundo.
- Grafo ponderado: Um grafo $G(V, E)$ é ponderado se existem valores numéricos associados a suas arestas ou nós.
- Grafo rotulados: Um grafo $G(V, E)$ é rotulado se existem atribuições associados a seus nós (tanto numéricas quanto alfabéticas).
- Grafo conexo: um grafo é conexo se for possível estabelecer um caminho de qualquer vértice para qualquer outro vértice de um grafo. Se for sempre possível estabelecer um caminho de qualquer vértice para qualquer outro vértice mesmo depois de remover

$k_\mu - 1$ vértices, então diz-se que o grafo é k-conexo. Note que um grafo é k-conexo se, e somente se, contém k_μ caminhos independentes entre qualquer par de vértices.

4 MÉTODO PROPOSTO

Neste capítulo é definido o conceito de novidade, descritos os métodos para a codificação do texto e imagem e para a fusão dos dados.

4.1 Definição de novidade

Neste trabalho, a novidade é proposta como sendo um evento que se destaca pela sua singularidade em relação ao tempo presente e passado. O tempo pode ser definido por janelas temporais, podendo ter sua medida temporal customizada, em intervalos de horas, dias e meses, etc. A Equação 4.1 apresenta o modelo matemático para classificar um evento em novidade ou não novidade:

$$f(w_{pr}, w_{pa}, t_{ac}) = \begin{cases} \text{Novidade}, & \text{Se } (t_{ac} \notin w_{pr}) \wedge (t_{ac} \notin w_{pa}) \\ \text{Não Novidade}, & \text{Caso contrário} \end{cases} \tag{4.1}$$

Se um evento t_{ac} recém-chegado do *stream* de dados não pertence a janela do presente w_{pr} ou a janela do passado w_{pa} e seu conteúdo/assunto (texto ou imagem) não está associado ou relacionado a qualquer outra ocorrência existente nessas janelas temporais, então essa ocorrência é identificada como uma novidade. O operador de não pertinência $\notin$ de um evento é verificado pelo algoritmo de aprendizado não supervisionado ou uma heurística. Por exemplo, se a janela temporal é de 4 horas, e uma notícia recém-chegada na janela temporal e diferente das outras for: “Cotação atual do dólar americano”, então provavelmente a notícia nova será rotulada como novidade. Em amostras temporais com tamanhos curtos as chances são baixas de ocorrem notícias similares. Mudando a janela temporal de 4 horas para 10 dias, a notícia cotação do dólar provavelmente deverá ser rotulada como não novidade porque sua amostra temporal já conterá notícias sobre cotação do dólar fornecido por algum outro usuário do Twitter.

Neste trabalho, um evento se refere a um *tweet*, mas pode ser usado para qualquer outro tipo de evento. O *tweet* é uma publicação feita por um usuário da rede social Twitter. A Figura 34 exemplifica um *tweet* postado, em que cada círculo colorido denota um metadado, como: fonte da postagem, título do *tweet*, autor, imagem, resumo ou texto do *tweet* e link.

Figura 34 – Exemplo de um *tweet* e seus metadados.

Autoria própria.

4.2 Qualidade da novidade

As Equações 4.2 e 4.3 caracterizam a qualidade da novidade que é medida pela audiência (MCQUAIL, 1997) em uma escala de 1 (menos importante) a 4 (muito importante). Um evento com audiência igual a 4 indica grande repercussão, enquanto um evento igual a 1 representa fraca repercussão na rede social. A variável st é definida pela similaridade entre eventos. Neste trabalho é usada a distância Euclidiana para calcular a distância entre os eventos. A variável rt é a quantidade de compartilhamento dos eventos e fc é a quantidade de interações, por exemplo, *likes* no caso do Twitter. A variável n é a quantidade total de eventos dentro de uma janela temporal.

$$quality(st, rt, fc) = Quartil(\frac{1}{n}\sum_{i=0}^{n-1} st_i + norm(rt_i) + norm(fc_i)) \tag{4.2}$$

Onde,

$$norm(x) = (x - min(x))/(max(x) - min(x)). \tag{4.3}$$

A escolha do intervalo (1-4) é devido o uso da medida de quartil. Um evento igual a 4 indica grande repercussão, enquanto um evento com qualidade igual a 1 representa uma reação fraca dos usuários na mídia social. A função *Quartil* divide todos os eventos dentro de uma janela temporal em quatro grupos, cada grupo contendo 25% dos eventos. O primeiro grupo contém os eventos com a audiência menor, enquanto o quarto grupo contém os eventos com a maior audiência entre todos os grupos.

A função *Quartil* foi escolhida porque é essencialmente um especificador de localização. A intenção é saber onde os eventos mais importantes estão localizados dentro da distribuição de todas as novidades e não-novidades detectadas. Entretanto, a função *Quartil* é inapropriada se houver muitos dados repetidos(LANGFORD, 2006). O conjunto de dados proposto neste trabalho tem poucas repetições de valores, então a função é adequada aos objetivos propostos. Além disso, a função *Quartil* é especialmente útil se os dados não são simetricamente distribuídos ou contém *outliers*.

Ainda há outras características importantes que a indicação do nível de qualidade pode beneficiar na detecção da novidade:

- Eliminação das novidades ruidosas. Há muitas notícias que são raras, mas não são importantes para propósitos jornalísticos, como memes, propaganda entre outros. Por isso, no contexto em que a novidade é uma notícia, é interessante analisar a audiência da novidade.

- Identificação de novas fontes alternativas. Nem sempre uma novidade será publicada em primeira mão por grandes mídias de notícias (como Reuters, BBC, etc.). Nota-se que atualmente o conteúdo gerado pelo usuário tem se tornado tão popular que notícias costumam aparecer primeiro fornecidas pelo usuário do que em outras mídias de maior destaque (PETROVIC et al., 2013). Essas fontes de notícias alternativas frequentemente são blogs, fóruns, Twitter, pinboard, etc. O trabalho em questão foca principalmente no Twitter por causa da sua natureza em tempo real, isso é, pessoas geralmente postam sobre eventos no momento que estão acontecendo, muitas vezes reportando antes das mídias convencionais. Exemplos mais proeminentes incluem a morte de Osama Bin Laden (TheGuardian, 2011), ou a notificação da morte de Michael Jackson (Karl, H. , 2010), que foram noticiados por cidadãos comuns antes de qualquer outra mídia de notícias.

4.3 Heurística de rotulagem semiautomática

Uma heurística foi implementada para verificar a similaridade entre os eventos relacionados dentro da janela temporal. Essa heurística de rotulagem semiautomática foi proposta para apoiar na tarefa de rotulação dos dados, já que não foram encontrados dados apropriados para a proposta, e está detalhada no Algoritmo 7. O rotulador é semiautomático, pois após a geração do rótulo por meio do algoritmo, ainda há uma etapa de conferência manual dos rótulos gerados.

O primeiro passo do algoritmo é iniciar a leitura de todos os eventos $AllEvents$, tendo o evento atual armazenado na variável t_{ac} e a data de criação do evento em $dtCreated$. O algoritmo inicializa variáveis cruciais, como $dtBegin$ e $dtEnd$, que são as datas iniciais e

Algoritmo 7 PseudoCódigoRotulagem

Entrada: AllEvents - Dados de entrada, dtCreated - Data de início, Windays - Tamanho da janela
Saída: Dados de entrada rotulado

1: $flag \leftarrow 0$
2: **while** Not *AllEvents* is EOF **do**
3: **if** $flag == 0$ **then** ▷ inicializando variáveis
4: $flag \leftarrow 1$
5: $dtBegin \leftarrow dtCreated$
6: $dtEnd \leftarrow dtBegin + Windays$
7: **end if**
8: **if** Not $(dtBegin \leq dtCreated \leq dtEnd)$ **then** ▷ *tweet* atual dentro da janela temporal
9: $dtBegin \leftarrow dtCreated$
10: $dtEnd \leftarrow dtBegin + Windays$
11: $w_{pa} \leftarrow w_{pr}$
12: $w_{pr} \leftarrow empty$
13: **end if**
14: $w_{pr} \leftarrow Add(t_{ac})$
15: $lbl \leftarrow Similar(t_{ac}, w_{pr}, w_{pa})$
16: $aud \leftarrow Audience(t_{ac}, w_{pr})$
17: **end while**
18: **return** *AllEvents*

finais da janela temporal. A variável $Windays$ é o tamanho da janela temporal, que neste trabalho é configurado em dias: 1, 5, 10 e 20 dias.

A escolha do tamanho das janelas está ligada a duas razões: contexto e ao volume de dados. Se o objetivo é descobrir as novidades do contexto de economia, possivelmente o tamanho das janelas temporais não serão customizadas em segundos ou minutos, mas em dias ou semanas, por exemplo: o recorde do dólar durante o dia ou semana. Aiello et al. (2013) observaram que o volume de dados sobre o tempo está ligado diretamente ao contexto, por isso a modelagem deste trabalho permite customizar o tamanho das janelas temporais de acordo com a necessidade do problema.

Também é importante observar o volume de dados dentro das janelas temporais, pois janelas curtas tendem a ter pouco volume de dados, causando o *overfitting* no treinamento. A seleção de dados deste trabalho optou em selecionar *tweets* que possuíam texto e imagem, por isso o volume de dados se mostrou menos abundante em janelas temporais muito curtas, caso fossem selecionados *tweets* apenas com texto.

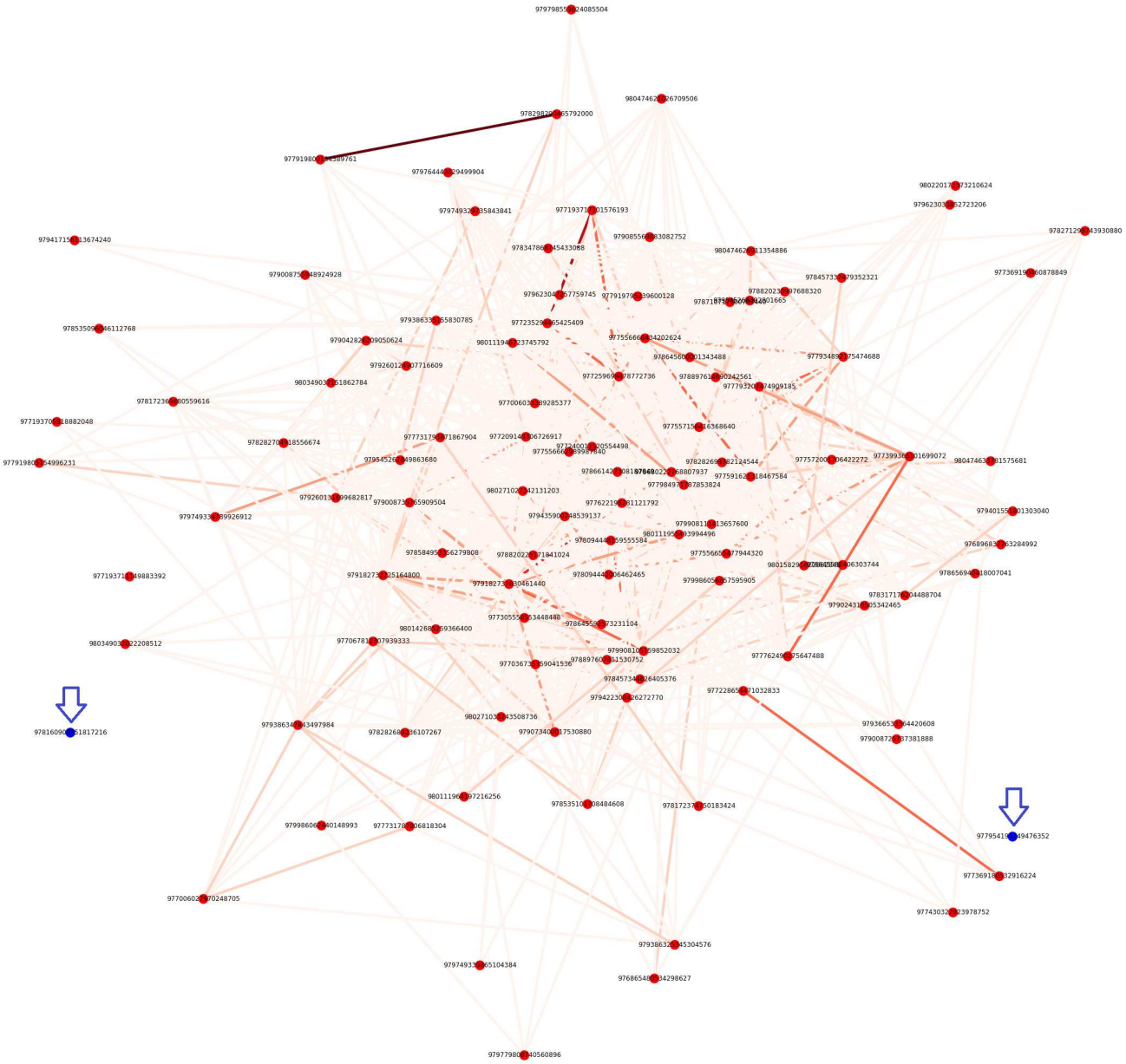
A data final é composta da data inicial mais o tamanho da janela temporal. Por exemplo: se o primeiro *tweet* ao entrar na janela tem a data inicial igual a 02/07/2019 e o tamanho da janela está configurado com 5 dias, então a data final será igual a 06/07/2019. A variável $flag$ igual zero indica que é a primeira vez no *loop*, e portanto é necessário inicializar variáveis da janela temporal.

Para cada passo no *loop*, o algoritmo verifica se o evento atual t_{ac} está dentro da janela temporal $dtBegin \leq dtCreated \leq dtEnd$. Se o evento atual não estiver dentro da janela temporal, então uma nova janela temporal é criada, caso contrário, as seguintes ações são realizadas: a variável t_{ac} é adicionada na lista de janelas temporais do presente w_{pr}, o rótulo *lbl* é retornado baseado na distância Euclidiana entre todos os eventos da janela temporal do passado w_{pa} e da janela temporal do presente w_{pr} e, finalmente, a audiência do evento atual t_{ac} em relação aos eventos do presente w_{pr} é calculada.

Como consequência da definição de novidade, a função de janelamento temporal tende a ser inversamente proporcional ao número de novidades detectadas, isso é, um grande período da janela temporal, implica em uma menor porcentagem de novidades em relação a quantidade de *tweets* dentro da janela. Por outro lado, um curto período da janela temporal, implica uma maior porcentagem de novidades detectadas. A hipótese para esse fenômeno não é a quantidade de janelas, mas a quantidade de dados dentro da janela. Se há menos amostras de dados dentro de uma janela, há uma chance menor de uma notícia se repetir, portanto se tornando uma notícia nova.

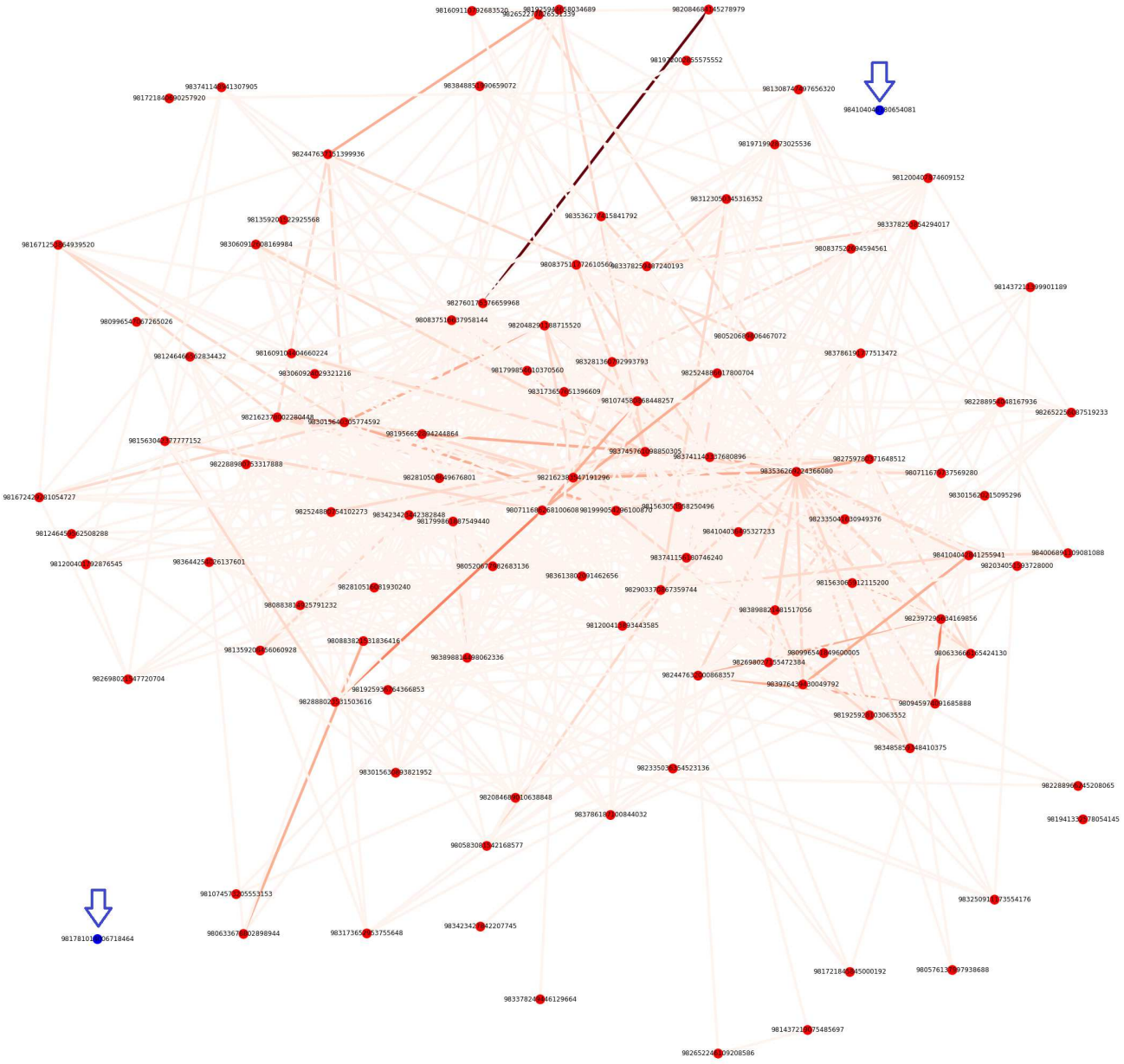
As Figuras 35 e 36 representam em forma de grafo a rotulação da janela temporal de 10 dias. Os vértices vermelhos indicam eventos que não são novidades e os azuis indicam novidades. O peso ou distância das arestas indicam o grau de similaridade entre os eventos, que é calculada usando distância Euclidiana. A distância Euclidiana ou norma L2 foi escolhida por ser mais usual na literatura de processamento de linguagem natural (HUANG, 2008). Uma outra opção seria utilizar a distância de Manhattan ou norma L1 que é mais robusta na detecção de *outliers* (KUMAR, 2017) (SCHUBERT et al., 2012) (VADIVEL; MAJUMDAR; SURAL, 2003).

Figura 35 – *Tweets* referentes ao intervalo 21/04/2018 a 01/05/2018. As setas destacam os eventos novidades, e que são representados pelos vértices em azul. As não novidades são representadas pelos vértices em vermelho.



Autoria própria.

Figura 36 – *Tweets* referentes ao intervalo 01/04/2018 a 11/04/2018. As setas destacam os eventos novidades, e que são representados pelos vértices em azul. As não novidades são representadas pelos vértices em vermelho.



Autoria própria.

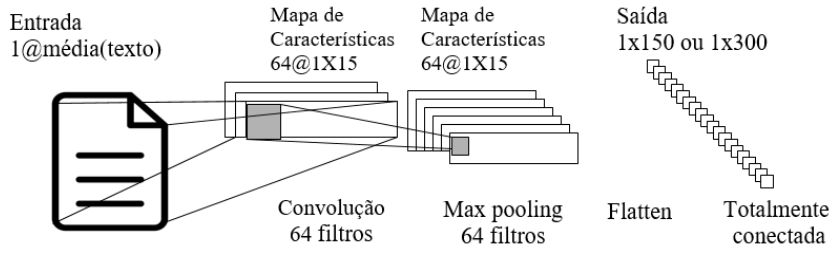
4.4 Codificação do texto e imagem

Um codificador (*encoder*) pode transformar o texto e a imagem em vetores de pesos de números reais (GOODFELLOW; YOSHUA; COURVILLE, 2016). Um *encoder* adequado é essencial para o desempenho de um classificador ou regressor (BENGIO;

COURVILLE; VINCENT, 2013) (ROSA et al., 2018). Portanto, os *encoders* usados neste trabalho são baseados em Vijay, Alex e Roberto (2017) [1] e Severyn e Moschitti (2015) [2]. A arquitetura de Severyn e Moschitti (2015) é baseada em camadas convolucionais que aprendem representações de textos curtos com o fim de relacionar pares de textos. A arquitetura de Vijay, Alex e Roberto (2017), conhecida como SegNet, é própria para segmentação semântica de imagens. As arquiteturas em questão foram escolhidas por serem apropriadas para os tipos de dados usados neste trabalho e apresentarem bons resultados na literatura.

Para este trabalho, a rede de codificação do texto, ilustrada na Figura 37, tem os seguintes parâmetros: duas camadas convolucionais 1-D, com 64 filtros de tamanho 1×15, com função de ativação ReLU, *maxpooling* 1-D de tamanho 1×7, seguido por um *dropout* de 0,25, e uma camada totalmente conectada de neurônios 1×150 ou 1×300, dependendo do tamanho escolhido para representar o texto. A dimensão de entrada desta rede é igual ao tamanho do dicionário da base, ou seja, igual ao número de palavras distintas.

Figura 37 – Rede de codificação do texto (*Encoder*).



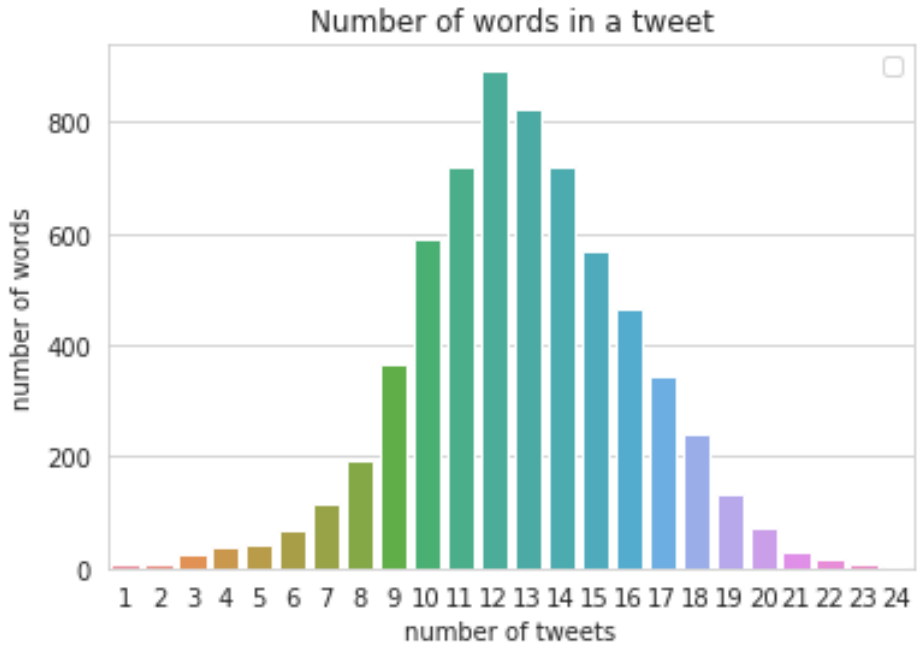
Autoria (SEVERYN; MOSCHITTI, 2015)

A rede usada foi pré-treinada com o Google News Dataset[3] (100 bilhões de palavras extraída do Google News e gera um modelo com o dicionário de 3 milhõe de palavras). Todo dicionário contém 14.425 palavras diferentes e 945 vetores (*embeddings*) nulos, e em média o *encoder* produziu um dicionário com 68 *tokens* singulares (palavras não repetidas) para a janela de 1 dias, 73 *tokens* para a janela de 5 dias, 155 *tokens* para a janela de 10 dias e 251 tokens para a janela de 20 dias. Além disso, foi considerado que o texto de entrada deve ter no mínimo três palavras. A rede tem um texto de entrada composta de no máximo 1000 palavras e uma rede com dimensão de saída de 1×150 a 1×300. A Figura 38 ilustra o número de palavras por evento no dicionário.

Há várias abordagens para codificar uma sentença em vetores, tais como: média de vetores, *one-hot*, etc. A abordagem escolhida neste trabalho é conhecida, em processamento de linguagem natural, como média vetorial. A seguir será detalhado os passos para a codificação do texto de treinamento utilizando vetores densos pré-treinados do Google.

[1] https://github.com/alexgkendall/SegNet-Tutorial
[2] https://github.com/zhangzibin/PairCNN-Ranking
[3] https://code.google.com/archive/p/word2vec/

Figura 38 – Distribuição do tamanho dos *tweets*.



Autoria própria.

1. **Pré-processamento:** O pré-processamento de texto é importante para que ruídos que possam atrapalhar o algoritmo sejam eliminados. Para isso, os seguintes processos são aplicados a base de treinamento.

   a) Todas as letras foram passadas para minúsculas (*lowercase*). Assim, por exemplo, se houver no texto *House* e *house*, elas serão consideradas iguais, como desejável.

   b) Remoção de *stopwords*. Essas palavras não agregam informação ao significado de um frase, como por exemplo preposições e conjunções. Se esse tipo de palavra for ignorado, é possível ainda compreender o significado da frase.

   c) Remoção de alguns caracteres especiais, como travessão e outros símbolos similares.

2. **Tokenização:** Neste passo, o texto é dividido em unidades significativas de texto chamadas *tokens*. Essas unidades devem ter sentido linguístico. Por exemplo, guarda-chuva é um *token* e *casa* é outro *token*. O resultado deste passo é uma lista de *tokens*, representados por *strings*, de todo o texto do treinamento.

3. **Mapeamento de *tokens* para um modelo de linguagem:** De posse das palavras da base de dados de treino, é possível mapeá-las para algum modelo de linguagem. Modelo de linguagem, de acordo com a área de Processamento de Linguagem Natural (PLN)(JURAFSKY, 2000), é como as palavras são representadas para o computador. Existem diversas formas de representação, como vetores densos ou *bag of words*.

Os vetores densos, também conhecidos como *word embeddings*, é uma forma mais moderna de representação em que cada *token* é representada por um vetor de número reais (MIKOLOV et al., 2013). Assim, foi escolhido *word embeddings* para representar a base de treino textual proposta neste trabalho. Além disso, foi escolhido usar um modelo de linguagem pré-treinado chamado de *Google Embeddings*.

As vantagens de utilizar um modelo de linguagem pré-treinado são a grande quantidade de palavras representadas e a representação mais precisa devido a grande base de treino do Google. Tais vantagens tendem levar para uma acurácia maior da rede.

Por fim, nesta fase de mapeamento, cada palavra da lista resultante foi mapeada da fase anterior para uma representação do tipo *word embedding*.

4. **Montagem da representação do *tweet*:** Esta é a última etapa para a codificação do texto de um *tweet* propriamente dito. Cada *tweet* contém um conjunto de palavras, em que cada palavra foi mapeada para um vetor, conforme passo anterior. Agora um conjunto de vetores é criado e por fim a média desses vetores é calculada, que resulta em apenas em um único vetor. Este vetor, que representa o *tweet*, é a representação que pode ser usada pelo computador.

A Figura 39 mostra um esquema de como essa representação é construída. No final é visto apenas um vetor, o qual representa todo o texto do *tweet*.

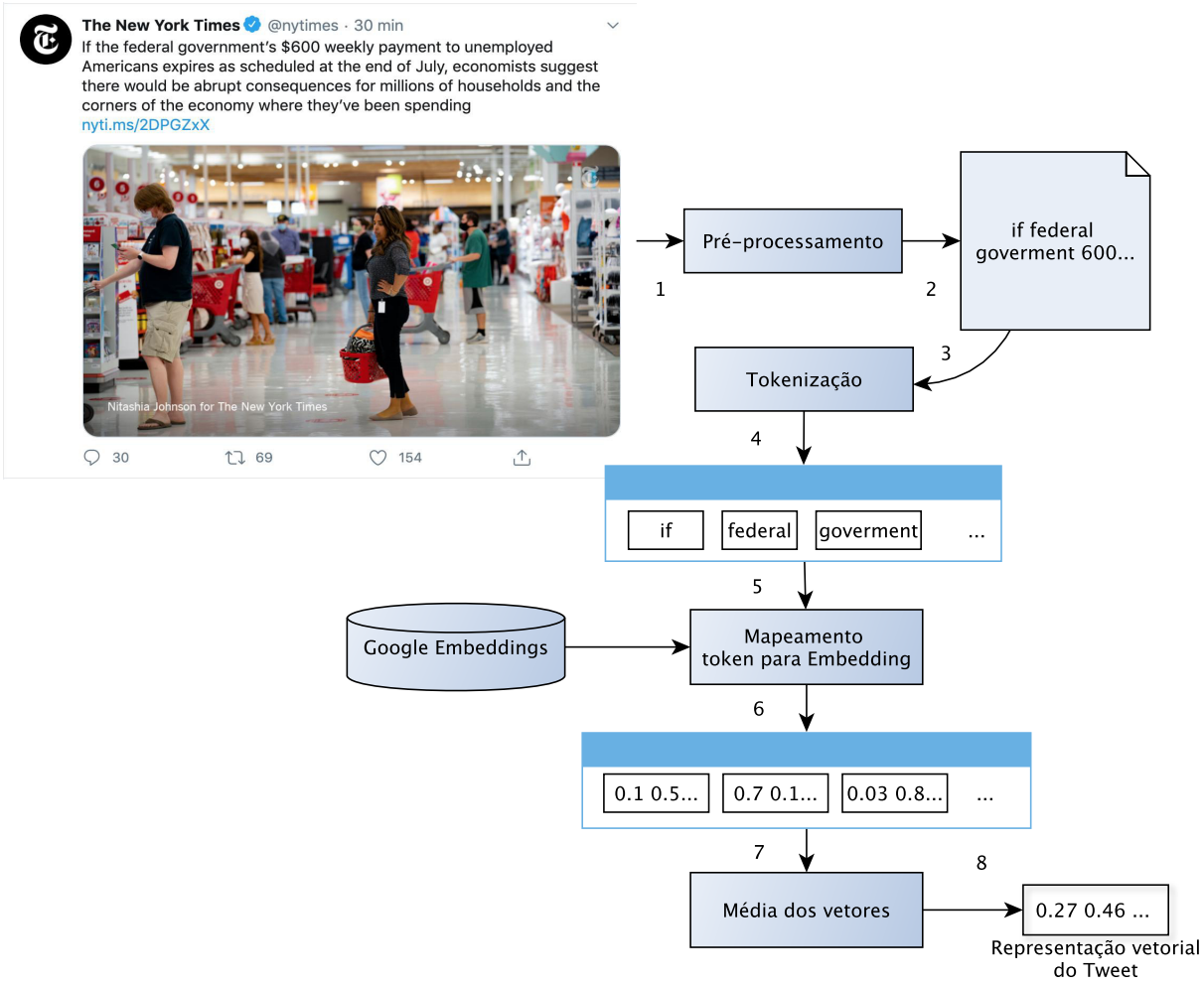
O modelo de linguagem representado pelo Google *embeddings* se dá por um outro processo que passa por fases de pré-processamento, tokenização, até o treinamento da representação propriamente dita. Essas representações resultam em vetores similares aos que vemos na Figura 39, onde cada dimensão do vetor possui um número real. Esta representação consegue capturar determinados conceitos semânticos das palavras, pois utiliza o contexto de uma dada palavra para inferir esse significado. Para ficar mais claro, são descritas as etapas utilizadas pelo Google para gerar seus *embeddings*(MIKOLOV et al., 2013) utilizando a base do Google News com 3 milhões de palavras.

1. **Pré-processamento:** Aqui todas letras são convertidas para minúsculas, símbolos especiais são removidos e *stopwords* são eliminadas.

2. **Tokenização:** Nesta etapa, a base de notícias do Google é dividida em *tokens*. O processo é similar ao utilizado na tokenização anterior, em que as unidades de texto significativas são separadas.

3. **Montagem do dicionário:** De posse dos *tokens* pré-processados, é necessário remover palavras pouco utilizadas. Essas palavras podem ser raras ou até mesmo conter erros ortográficos, portanto contribuem pouco para para um modelo de linguagem. Nesta etapa, 3 milhões de palavras foram selecionadas para a construção

Figura 39 – Transformação de um *tweet* para vetor denso, i.e., representação em *word embedding*.



Autoria própria.

do dicionário do modelo de linguagem. Assim, o modelo de linguagem conseguirá representar apenas as palavras contidas em seu dicionário.

4. **Treinamento:** Por fim, as palavras do dicionário serão transformadas em representações chamadas *word embeddings*. Nesta fase será feito o treinamento de um rede neural. No treinamento passamos pelas seguintes fases:

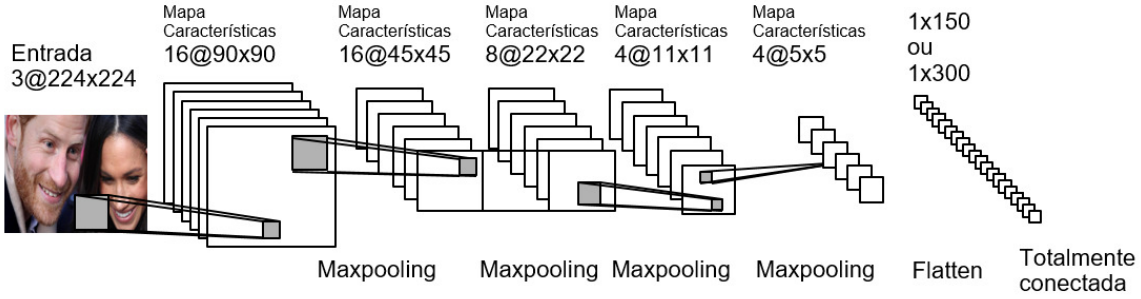
   a) **Enviar entrada:** a rede neural de codificação de texto recebe uma matriz como entrada, em que cada linha da matriz representa uma palavra de uma janela em torno de uma palavra w. Uma janela são as palavras que circundam w e cuja representação deseja ser aprendida. Assim, uma janela é o contexto da palavra w. A matriz de entrada é representada na codificação *one-hot*, portanto a dimensão representada pela palavra será 1, e as demais dimensões serão 0. A saída da rede tentará prever o *embedding* da palavra w.

b) **Backpropagation:** Após a entrada ser processada pelos neurônios da rede, é gerado um *embedding* intermediário. O custo desta representação é computada e então a partir do custo, os pesos da rede neural são ajustados através do mecanismo de *backpropagation*. Essa etapa é executada diversas vezes até que o custo da representação seja aceitável.

c) **Gerar a saída:** Por fim a rede gera um vetor representando a palavra do w. Esse vetor gerado pelo é similar ao vetor da Figura 29 na Seção 3.12, ou seja, são vetores de números reais de tamanho 1×150 ou 1×300 que resultaram da combinação da entrada com a matriz de pesos da camada oculta. Para gerar o vetor que representa o *tweet* é realizado a média vetorial dos *embeddings* das palavras do *tweet*.

A rede de codificação das imagens tem os seguintes parâmetros: uma camada convolucional 2-D com 16 filtros de tamanho 90×90, função de ativação ReLU, *maxpooling* 2-D com janela de tamanho 2×2, outra camada com 16 filtros de 45×45, com função de ativação ReLU, *maxpooling* 2-D com janela de tamanho 2×2, mais uma camada com 8 filtros de tamanho 22×22, função de ativação ReLU, *maxpooling* 2-D com janela de tamanho 2×2, uma quarta camada convolucional com 4 filtros de tamanho 11×11, função de ativação ReLU, *maxpooling* 2-D com janela de tamanho 2×2 e uma última camada convolucional com 4 filtros de tamanho 5×5, com função de ativação ReLU, *maxpooling* 2-D com janela de tamanho 2×2 seguido por um dropout de 0,25, uma camada totalmente conectada com neurônios 1×150 ou 1×300. A saída é um vetor de números reais de tamanho 1×150 ou 1×300. A rede está ilustrada na Figura 40 (VIJAY; ALEX; ROBERTO, 2017).

Figura 40 – Rede de codificação da imagem (*Encoder*).



Autoria (VIJAY; ALEX; ROBERTO, 2017)

4.5 Fusão do texto e da imagem

A fusão é um processo que combina a entrada ou a saída de múltiplos classificadores a fim de obter melhor desempenho na classificação. Neste trabalho foram usadas duas

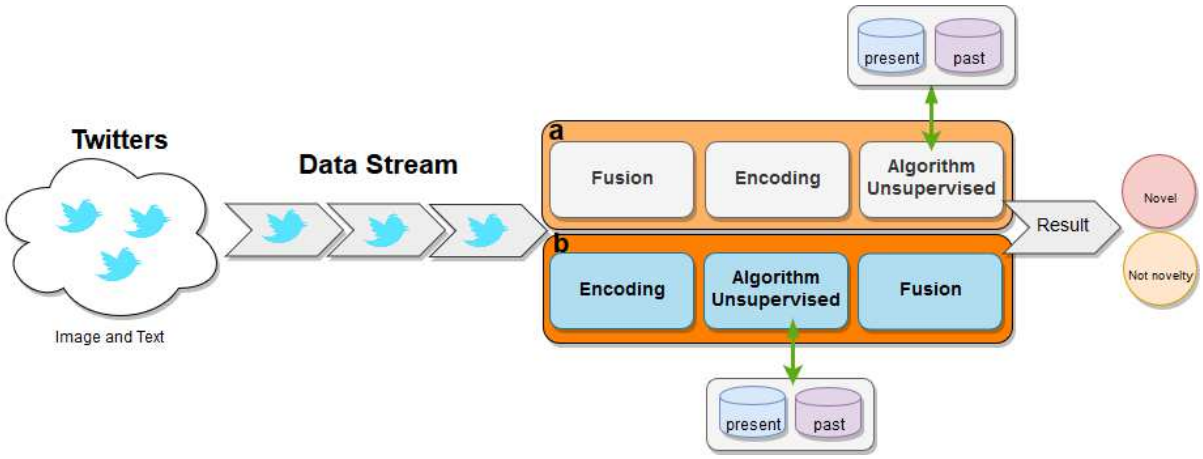
abordagens diferentes: uma fusão na entrada do classificador usando a rede *MASK R-CNN* (HE et al., 2017) e uma fusão na saída denominada AOM (*Average-of-Maximum*) (AGGARWAL; SATHE, 2015). O AOM é um método de *ensemble* que combina a saída de múltiplos classificadores.

Neste trabalho foi desenvolvido um novo método para fusão da entrada. O novo método usa uma rede convolucional MASK R-CNN (HE et al., 2017) treinada com a base de dados COCO (LIN et al., 2014) para reconhecer objetos em imagens, como mostrado nas Figuras 42, 43, 44 e 45. Os rótulos resultantes com probabilidades maiores que 0,7 são adicionados ao texto da notícia para agregar e suplementar informação. A partir disso, o texto é codificado e, por fim, um algoritmo não supervisionado (como um daqueles apresentados no Capítulo 3) identifica se a informação é novidade ou não novidade. Essa abordagem, ilustrada na Figura 41 (a), é uma fusão da entrada no nível da estrutura dos dados (imagem + texto), que transforma a imagem em uma única estrutura de dados textual. Uma fusão da entrada tem vantagem sobre a fusão da saída, porque, a fusão da saída combina probabilidades que geralmente precisam dos votos dos estimadores, e que podem estar incorretos quando se usa apenas texto ou imagem como entrada. A fusão proposta não elimina a possibilidade de usar a fusão na saída como fusão adicional.

A segunda abordagem, ilustrada na Figura 41 (b), usa a fusão AOM (*Average-of-Maximum*) (AGGARWAL; SATHE, 2015). Essa fusão distribui a tarefa de detecção de novidades em m estimadores, e os estimadores são divididos em m/q subgrupos de q estimadores cada. Primeiro, a maximização é usada em cada um dos subgrupos de q estimadores, sendo que a pontuação é a média de m/q subgrupos. Por exemplo, se for usado $m = 20$ classificadores para imagem e $m = 20$ para classificadores de texto, então os classificadores são divididos em 10 subgrupos de 2 classificadores cada. Primeiro, o maior valor é usado em cada subgrupo de 2 classificadores, e então as pontuações (probabilidades) de saída são as médias de todos os subgrupos. A técnica mostrou bons resultado em algoritmos de detecção de *outlier*, que é relacionado a detecção de novidades porque ambas as técnicas empregam algoritmos não supervisionados. Neste trabalho, todos os estimadores para o AOM são algoritmos de aprendizado não supervisionados. Outros métodos de fusão foram testados, como: MOM *Maximum of Maximum*,MOA *Maximum of Average*, entre outros, mas foi selecionado AOM por apresentar o melhor resultado.

O Algoritmo 8 descreve em passos como a abordagem ilustrada na Figura 41 funcionaria em modo *stream*.

Figura 41 – O *pipeline* das duas arquiteturas propostas para detecção de novidades em tempo real. A arquitetura na parte (a) funde o texto e a imagem usando a nova abordagem proposta por este trabalho usando o *Dataset* COCO e a rede convolucional *MASK R-CNN*. Primeiro, a fusão dos dados é realizada a partir dos rótulos gerados pela rede *MASK R-CNN*. Após, o resultado da fusão é codificado e finalmente a novidade é classificada usando algoritmos não supervisionados. A arquitetura na parte (b) realiza a codificação separadamente, classifica com algum algoritmo não supervisionado e finalmente realiza a fusão dos dados com o algoritmo AOM.



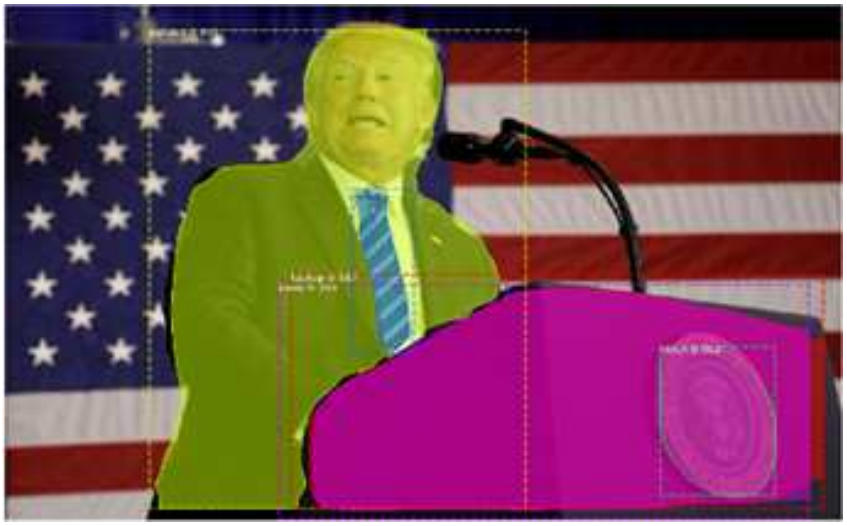
Autoria própria.

Algoritmo 8 Pseudo Código da Arquitetura Proposta

Entrada: User - Usuário do *tweet*, Pass - Senha do *tweet*

Connect ← *OpenConnectionTwitter(User,Pass)* ▷ Abre conexão *Twitter*
while *(true)* **do**
  try
    Tweet ← *ReadStream(User,Pass)* ▷ Obtém o *tweet*
    if *Tweet* **is** *Valid()* **then**
      TaggingLabel(Tweet) ▷ Heurística rotular o *tweet*
      PipelineA(Tweet) ▷ *Pipeline* A de classificação *tweet*
      PipelineB(Tweet) ▷ *Pipeline* B de classificação *tweet*
    end if
  catch Connection exception ▷ Erro ao abrir conexão *Twitter*
    Connect ← *OpenConnectionTwitter(User,Pass)*
  end try
end while

Figura 42 – Exemplo de uma imagem de um *tweet* em que a rede pré-treinada *MASK R-CNN* reconhece os seguintes objetos em cena: *person* (0,999), *tie* (0,998), *clock* (0,952), *book* (0,346) e um *laptop* (0,917).

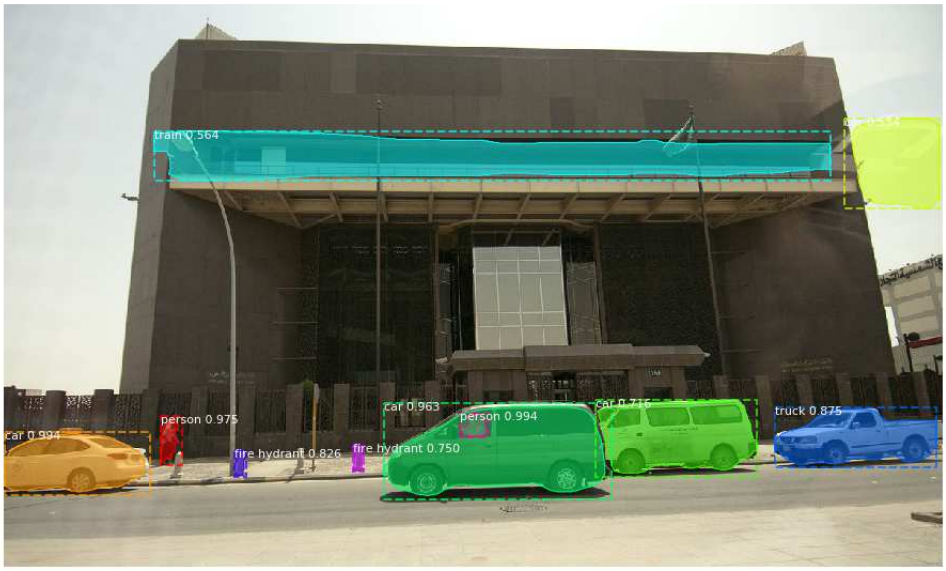


Autoria (LIN et al., 2014)

Figura 43 – Exemplo de uma imagem de um *tweet* em que a rede pré-treinada *MASK R-CNN* reconhece os seguintes objetos em cena: *person* (0,996), *person* (0,999) e *person* (0,994).

Autoria (LIN et al., 2014)

Figura 44 – Exemplo de uma imagem de um *tweet* em que a rede pré-treinada *MASK R-CNN* reconhece os seguintes objetos em cena: *train* (0,564), *car* (0,994), *person* (0,975), *fire hydrant* (0,750), *car* (0,963), *person* (0,994), *car* (0,716), *truck* (0,875) e *train* (0,564).



Autoria (LIN et al., 2014)

Figura 45 – Exemplo de uma imagem de um *tweet* em que a rede pré-treinada *MASK R-CNN* reconhece os seguintes objetos em cena: *boat* (0,632), *boat* (0,979), *boat* (0,995), *boat* (0,521), *chair* (0,0687), *boat* (0,973) e *boat* (0,986).

Autoria (LIN et al., 2014)

5 EXPERIMENTOS

Neste capítulo são detalhados a métrica utilizada, a configuração do ambiente de desenvolvimento, os experimentos e a análise dos resultados.

5.1 Métrica

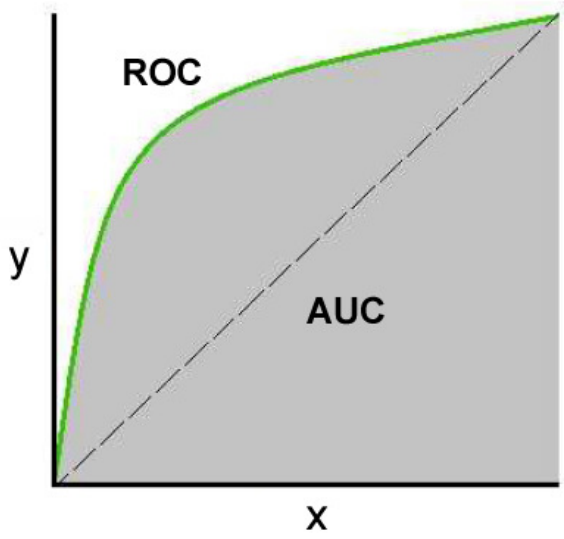
A questão chave surge como medir de forma precisa um algoritmo de detecção de novidades. Isso é uma tarefa difícil, pois novidades assim como *outliers*, por definição, são raros. Além disso, significa também que os rótulos dos dados são binários, ou seja, sendo novidades ou não novidades, tendo dessa forma um desbalanceamento dos dados favorecendo a classe de não novidades (AGGARWAL, 2016).

Diante das características da tarefa de detecção de novidade, que são classificação binária e desbalanceamento dos dados, foi utilizada neste trabalho a métrica ROC AUC (*Receiver Operating Characteristic - Area Under the Curve*). Geometricamente, a ROC é uma curva de pares "x" e "y" que relaciona a taxa de verdadeiros positivos (TVP) contra a taxa de falsos positivos (TFP), conforme Figura 46. A taxa de verdadeiros positivos é o percentual dos eventos que verdadeiramente foram reportados como novidades, também conhecido como sensibilidade. A taxa de falsos positivos é o percentual dos eventos que falsamente foram reportados como novidades, também conhecido como especificidade. Cada ponto neste espaço ROC corresponde a um par de valores TVP e TFP obtido pelo uso de um valor de limiar na saída de um classificador. A variação deste limiar, desde um valor baixo até um valor alto, produz a curva ROC (BRAGA, 2000).

Para comparar os classificadores, é possível sintetizar o desempenho da ROC para um único valor escalar que representa o desempenho esperado. Um método comum é calcular a área sob a curva ROC, abreviada para AUC. Como a AUC é uma parte da área de uma unidade quadrada, seu valor sempre estará entre 0 e 1. A AUC tem uma propriedade estatística importante: a AUC de um classificador é equivalente à probabilidade de o algoritmo classificar uma instância positiva escolhida aleatoriamente maior do que uma instância negativa escolhida aleatoriamente. Isso é equivalente ao teste de postos de Wilcoxon e também está intimamente relacionada ao coeficiente de Gini. A AUC é a região marcada em cinza na Figura 46 (FAWCETT, 2004).

A ROC ajuda a decidir o melhor limiar, por meio da TVP e TFP que pode ser plotado em um gráfico. As Equações 5.1 e 5.2 detalham a TVP e TFP (SOKOLOVA;

Figura 46 – Curva ROC relacionando a taxa de verdadeiros positivos (TVP) contra a taxa de falsos positivos (TFP), em que o eixo x é TFP e o eixo y é TVP



Autoria (Sarang Narkhede, 2018)

LAPALME, 2009):

$$TVP = \frac{VP}{VP + FN} \tag{5.1}$$

$$TFP = \frac{VN}{VN + FP} \tag{5.2}$$

A Tabela 5 ilustra uma matriz de confusão[1] com a definição de cada variável das Equações 5.1 e 5.2 (SOKOLOVA; LAPALME, 2009)

Tabela 5 – Matriz de confusão para a classificação de novidade ou não novidade

Classe	**Classificado Novidade**	**Clas. como Não Novidade**
Novidade	Verdadeiro Positivo (VP): evento reportado como Novidade É NOVIDADE	Falso Negativo (FN): evento rep. como Não Novidade É NOVIDADE
Não Novidade	Falso Positivo (FP): evento reportado como Novidade NÃO É NOVIDADE	Verdadeiro Negativo (VN): evento rep. como Não Novidade É NÃO NOVIDADE

(SOKOLOVA; LAPALME, 2009) afirmam que a curva ROC é mais apropriada para avaliação de bases de dados desbalanceadas por ser mais sensíveis do que outras métricas, tais como precisão e *recall*.

[1] Uma matriz de confusão é um formato específico de tabela que permite visualizar o desempenho de um algoritmo em termos de verdadeiros positivos e negativos; e falsos positivos e negativos.

5.2 Base de dados

Neste trabalho foi escolhido trabalhar com uma base de dados composta por postagens da rede social *Twitter*, por possuir textos e imagens livres para coleta de dados em um fluxo contínuo de dados. Entretanto, apesar de um grande número de bases de dados de *tweets* estarem disponíveis na literatura, as seguintes deficiências são observadas para o contexto desta pesquisa:

- A maioria das bases de dados são apresentadas em um contexto particular, isso é, os *tweets* são filtrados usando palavras-chave.
- A técnica proposta é baseada em janelas temporais, e uma mudança no rótulo em função de um intervalo de tempo. Encontrar bases de dados com este tipo de rotulação mostrou ser uma tarefa difícil. Os rótulos são essenciais para medir a efetividade da técnica.
- A técnica proposta requer que cada *tweet* deva ter texto e imagem para avaliar a robustez. Embora a técnica proposta possa funcionar só com textos ou só com imagens, um dos objetivos deste trabalho é avaliar o quanto a identificação de novidades pode ser melhorada ao combinar ambos tipos de dados. Portanto, é imprescindível ter uma base de dados cujos *tweets* tenham textos e imagens. Durante a pesquisa de dados não foram encontradas bases de dados públicas de *tweets* que tenham tanto textos quanto imagens, já que grande parte das pesquisas trabalha só com texto.

Como parte da evidência dos argumentos listados anteriormente, foi realizada a investigação de algumas bases de dados da literatura que poderiam fazer parte do *benchmarking* deste trabalho. Para que a base de dados seja útil no contexto desta pesquisa é necessário ter: (i) ID do *tweet* - É possível verificar se o *tweet* possui texto ou imagem. (ii) Rótulo - Não é necessário que seja uma base de dados aplicada a detecção de novidades, mas que tenha algum rótulo para que possa otimizar e medir a eficiência da técnica proposta. Dessa forma, a Tabela 6 apresenta uma lista das bases de dados investigadas.

Tabela 6 – Bases de dados da literatura.

Base de dados	**Total**	**Texto**	**Imagem**
Sentiment140 (GO; BHAYANI; HUANG, 2009)	1600000	1600000	0
Hate speech (MONDAL; SILVA; BENEVENUTO, 2017)	20305	20305	641
Rumor (KWON; CHA; JUNG, 2016)	34390	34390	0
Ofessive language (DAVIDSON et al., 2017)	14508	14508	0

Para superar essa limitação, 27,494 *tweets* criados no período 2014-2019 foram colecionados de diferentes fontes de notícias: BBC, BBC breaking News, CNN, Reuters,

NYtimes, etc, e rotulados pelo Algoritmo 7. Os rótulos foram manualmente verificados, com o objetivo de avaliar a eficácia do algoritmo. A base de dados e código fonte estão disponíveis para livre *download*[2].

5.3 Configuração dos experimentos

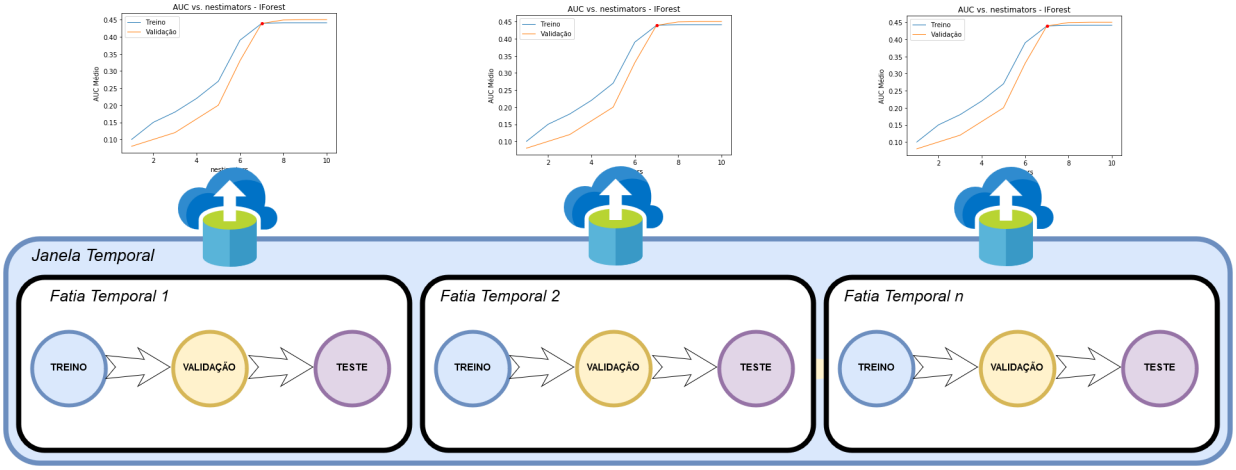
A arquitetura proposta neste trabalho foi desenvolvida usando Python 3.6.6, Keras 2.2.2, PyOD 0.5.9 para algoritmos não supervisionados e Anaconda 4.5.11 como ambiente virtual. Todos os experimentos foram realizados em um computador equipado com uma placa de vídeo NVIDIA 1080 GPU 8G, um processador Intel i7-26000K CPU 3.4 Ghz, e 32GB RAM, executando sobre o sistema operacional Ubuntu 18.04 LTS.

A base de dados foi fatiada em períodos regulares de tempo, que estão configurados em 1, 5, 10 e 20 dias. Cada fatia temporal contém amostras de treino, validação e teste, de forma que o algoritmo é treinado e validado com as amostras das janelas $t-1$ e t, e testado com as amostras da janela $t+1$, ou seja, as amostras usadas no modelo mudam conforme o fluxo de chegada dos dados. Também conhecido na literatura como aprendizado *online* baseado em deslizamento de janelas (IZZELDIN; ASIRVADAM; SAAD, 2011). Para um treinamento adequado, a janela de treino deve possuir 70% das amostras totais entre as janelas temporais $t-1$ e t. Se a quantidade de amostras for inferior a 50 *tweets*, então é realizada uma concatenação com as janelas do passado a fim de que a janela de treino alcance as 50 amostras. As outras 30% das amostras são usadas para validação e as amostras da janela t+1 são usadas para teste.

Por exemplo, seja a configuração da janela temporal igual 1 dia, então: a primeira fatia temporal contém *tweets* referentes aos dias 01/01/2017 e 02/01/2017 para treinamento e validação, e *tweets* do dia *03/01/2017* na fase de teste. A segunda fatia temporal contém *tweets* dos dias 02/01/2017 e 03/01/2017 para a fase de treinamento e validação, e *tweets* do dia *04/01/2017* na fase de teste, e assim por diante para cada fatia temporal. Supondo que na segunda fatia temporal, a janela de treino possua menos de 70% das amostras entre as janelas $t-1$ e t, então a janela do dia *02/01/2017* será concatenada com as janelas do passado, como a janela do dia *01/01/2017*. A Figura 47 esquematiza o processo de treinamento, validação e teste. Algumas fatias temporais são concatenadas com as fatias anteriores, devido ao pequeno número de *tweets* presentes.

[2] https://github.com/martatcfa/novelty

Figura 47 – Esquema do treinamento, validação e teste na janela temporal. Na fase de treinamento e validação é possível obter o melhor parâmetro por meio da curva de aprendizado.



Autoria própria.

5.4 Resultados

Os experimentos foram realizados com as técnicas de fusão MASK R-CNN e AOM. Além disso, com o intuito de verificar a robustez do método de fusão, foi inspecionado os algoritmos em relação a usar somente texto e somente imagem, como entrada para os algoritmos não supervisionados. A métrica AUC (*Area Under Curve*) foi utilizada para avaliar os resultados.

A seleção dos melhores parâmetros dos algoritmos não supervisionados é realizada por meio da dinâmica da curva de aprendizado na fase de treino e validação, e que pode apresentar os seguintes comportamentos: *underfitting*, *overfitting* ou ajuste adequado. Como cada fatia temporal possui um melhor parâmetro, os resultados na Tabela 7 apresentam o melhor parâmetro médio. A primeira coluna da tabela informa o nome do algoritmo, a segunda coluna o parâmetro algoritmo: k para o algoritmo kNN, *nestimators* para o algoritmo *IForest* e *FBagging*, *nbins* para o algoritmo *HBOS* e Épocas para algoritmo *Autoencoders*. A terceira coluna da tabela informa o tamanho da janela temporal em dias. A quarta coluna da tabela mostra os melhores parâmetros médios selecionados para a janela para cada tipo de dado ou método de fusão de dados.

Tabela 7 – Valores médios selecionados para os parâmetros dos algoritmos de aprendizado.

Algoritmo	Parâmetro	Janela	Melhor
kNN	k	1	3 texto
			7 imagem
			5 AOM
			4 MASK
		5	4 texto
			9 imagem
			7 AOM
			5 MASK
		10	3 texto
			6 imagem
			7 AOM
			5 MASK
		20	4 texto
			7 imagem
			5 AOM
			4 MASK
IForest	$nestimators$	1	9 texto
			12 imagem
			11 AOM
			9 MASK
		5	7 texto
			11 imagem
			9 AOM
			8 MASK
		10	6 texto
			8 imagem
			5 AOM
			7 MASK
		20	13 texto
			15 imagem
			14 AOM
			11 MASK
HBOS	$nbins$	1	4 texto
			5 imagem
			5 AOM
			4 MASK

Algoritmo	Parâmetro	Janela	Melhor
		5	4 texto 6 imagem 4 AOM 5 MASK
		10	6 texto 10 imagem 8 AOM 7 MASK
		20	4 texto 9 imagem 6 AOM 5 MASK
FBagging	*nestimators*	1	6 texto 7 imagem 7 AOM 5 MASK
		5	5 texto 10 imagem 8 AOM 7 MASK
		10	7 texto 8 imagem 6 AOM 7 MASK
		20	10 texto 12 imagem 12 AOM 11 MASK
Autoencoder	*epochs*	1	4 texto 6 imagem 7 AOM 5 MASK
		5	5 texto 8 imagem 7 AOM 3 MASK

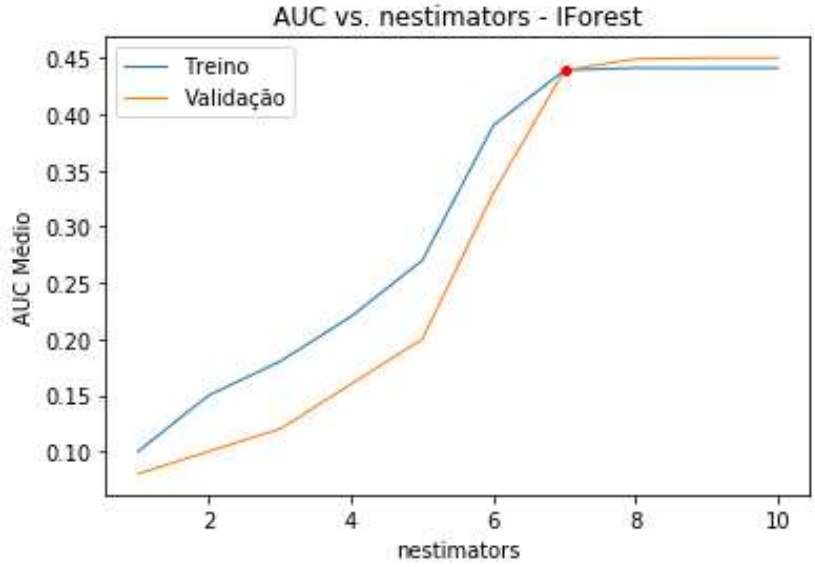
Algoritmo	Parâmetro	Janela	Melhor
		10	3 texto 5 imagem 7 AOM 3 MASK
		20	10 texto 8 imagem 13 AOM 11 MASK
LSTM	*epochs*	1	7 texto 9 imagem 7 AOM 12 MASK
		5	6 texto 12 imagem 5 AOM 10 MASK
		10	3 texto 11 imagem 10 AOM 7 MASK
		20	5 texto 9 imagem 4 AOM 13 MASK

Os outros parâmetros dos algoritmos não citados na calibragem neste trabalho foram ajustados de acordo com a literatura de referência de cada algoritmo: kNN (ANGIULLI; PIZZUTI, 2002), HBOS (GOLDSTEIN; DENGEL, 2012), *Feature Bagging* (LAZAREVIC; KUMAR, 2005), Autoencoder (CHEN et al., 2017), *Isolation Foresting* (LIU; TING; ZHOU, 2012) e LSTM (RANJAN et al., 2018). Esses parâmetros foram avaliados com outros valores diferentes, mas o treino/teste mostrou que a calibragem recomendada pela literatura apresentou melhores resultados.

As Figuras 48 a 53 exemplificam a obtenção do melhor parâmetro para a nova técnica usando MASK R-CNN com janela de 10 dias, cuja configuração produziu o melhor

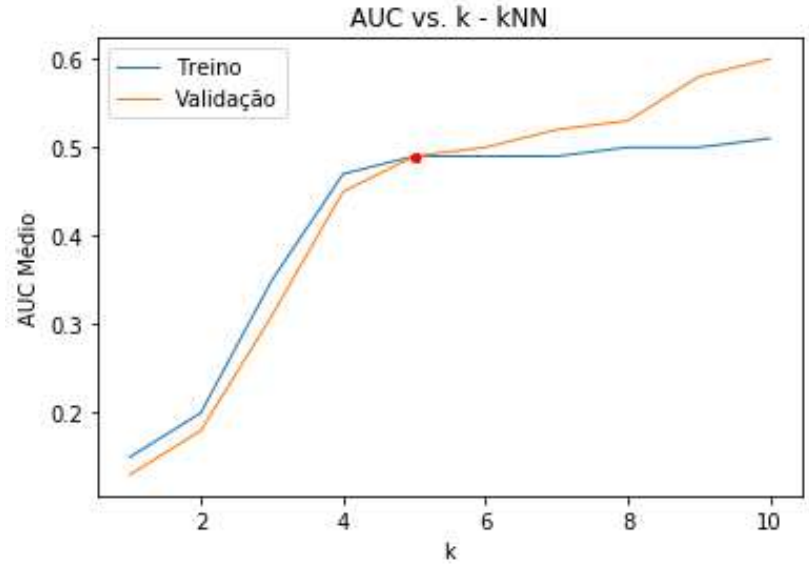
AUC médio dentre todas as outras janelas. O ponto em vermelho destaca o melhor parâmetro para o algoritmo em questão.

Figura 48 – Curva de aprendizado média para o algoritmo *IForest*. O melhor parâmetro *nestimators* médio para a nova técnica usando MASK R-CNN com janela 10 dias.



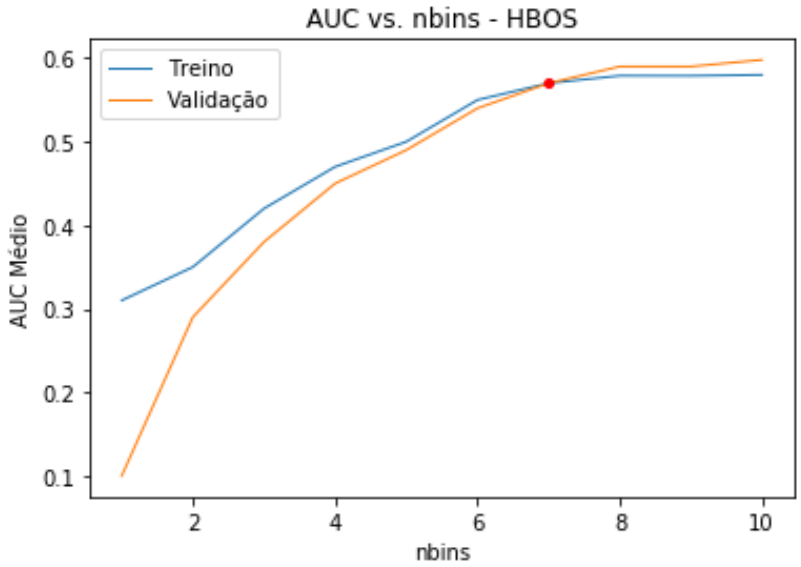
Autoria própria.

Figura 49 – Curva de aprendizado para o algoritmo kNN. O melhor parâmetro k médio para a nova técnica usando MASK R-CNN com janela de 10 dias.



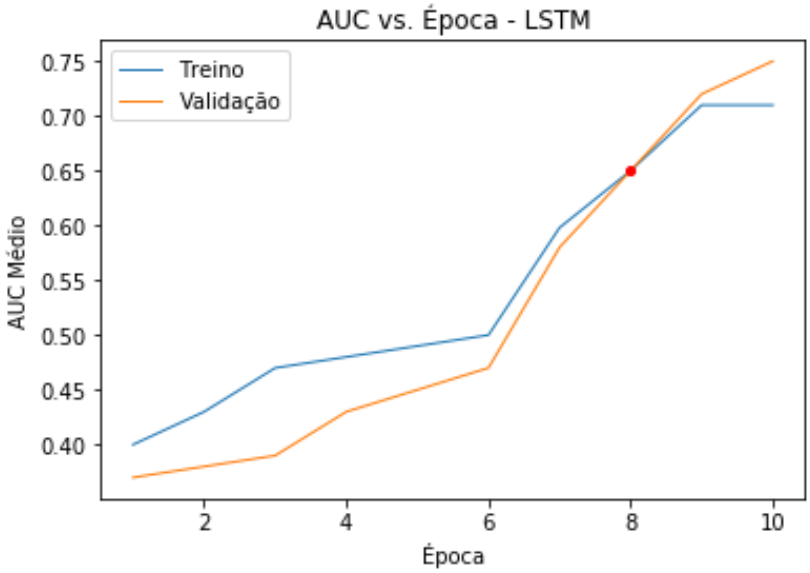
Autoria própria.

Figura 50 – Curva de aprendizado para o algoritmo HBOS. O melhor parâmetro *nbins* médio para a nova técnica usando MASK R-CNN com janela de 10 dias.



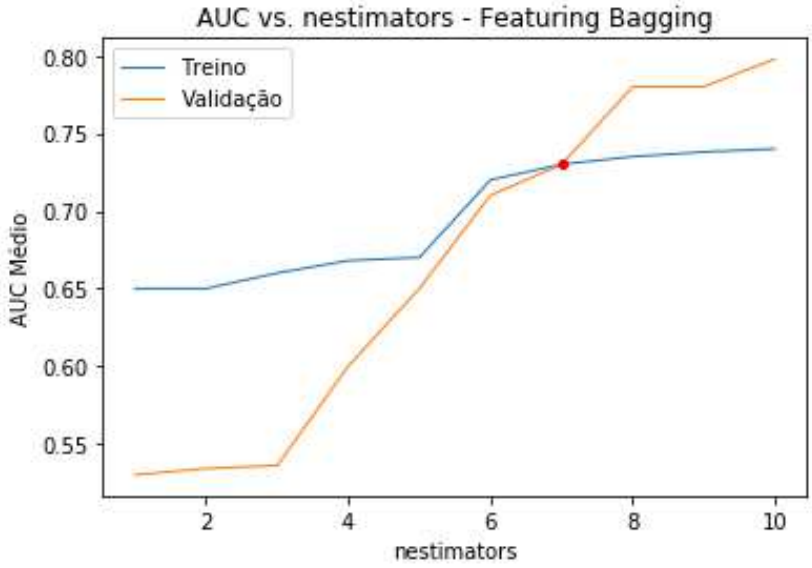
Autoria própria.

Figura 51 – Curva de aprendizado para o algoritmo LSTM. O melhor parâmetro Época médio para a nova técnica usando MASK R-CNN com janela de 10 dias.



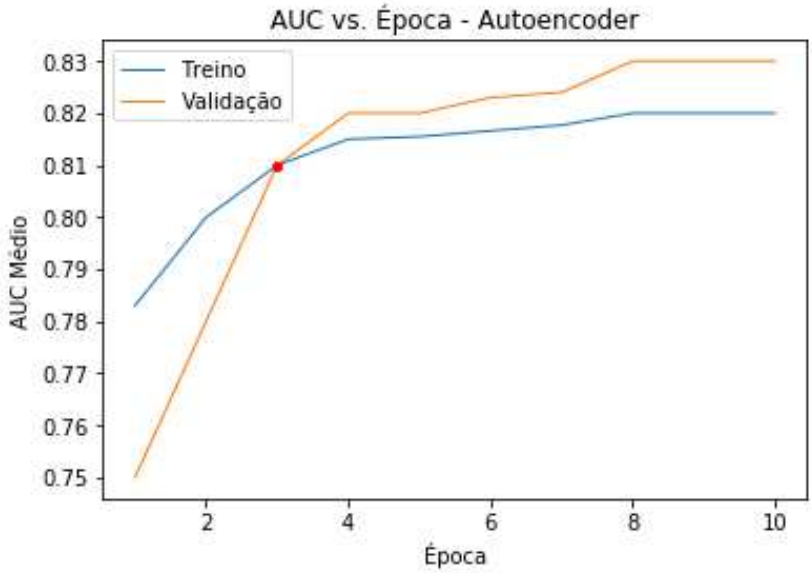
Autoria própria.

Figura 52 – Curva de aprendizado para o algoritmo *FBagging*. O melhor parâmetro *nestimators* médio para a nova técnica usando MASK R-CNN com a janela de 10 dias.



Autoria própria.

Figura 53 – Curva de aprendizado média para o algoritmo Autoencoder. O melhor parâmetro Época médio para a nova técnica usando MASK R-CNN com janela de 10 dias.



Autoria própria.

Os algoritmos foram executados 15 vezes, e o resultado final é a média das execuções, como apresentados nas Tabelas de 8 a 11 para os diferentes tamanhos de janela temporal. Os algoritmos do estado da arte de aprendizado não supervisionado também foram avaliados

pela influência do tamanho do vetor de entrada (texto ou imagens codificados), com tamanhos de 150 e 300.

Nas Tabelas de 8 a 11, a coluna texto mostra os resultados da classificação somente com texto. A coluna imagem mostra os resultados da classificação somente com imagem. A coluna AOM mostra os resultados usando a fusão de texto e imagem na saída, proposta por (AGGARWAL; SATHE, 2015). A coluna MASK mostra os resultados usando a nova abordagem utilizando a rede MASK R-CNN e a base de dados COCO como treinamento dela. Os valores destacados em vermelho são os melhores resultados alcançados para o tamanho do vetor de entrada do algoritmo.

Tabela 8 – Resultados da métrica ROC AUC para janela temporal de 1 dia.

Janela de 1 dia								
	Vetor de entrada = 150				Vetor de entrada = 300			
Algoritmo	Texto	Imagem	AOM	MASK	Texto	Imagem	AOM	MASK
HBOS	0,4124	0,4007	0,4358	**0,4794**	0,3982	0,3802	0,4214	**0,4561**
desvio padrão	0,0046	0,0053	0,0047	0,0149	0,0277	0,0216	0,0225	0,0082
kNN	0,3357	0,3038	0,3789	**0,4115**	0,3102	0,2986	0,3712	**0,3999**
desvio padrão	0,0258	0,0235	0,0206	0,0193	0,0037	0,0203	0,0216	0,0189
IForest	0,2989	0,2844	0,3077	**0,3348**	0,1855	0,1842	0,1966	**0,2205**
desvio padrão	0,0050	0,0219	0,0214	0,0042	0,0226	0,0837	0,0194	0,0122
Featuring Bagging	0,5338	0,5237	0,5446	**0,5558**	0,5056	0,4946	0,5087	**0,5182**
desvio padrão	0,0245	0,0086	0,0231	0,0212	0,0278	0,0177	0,0184	0,0041
Autoencoder	0,5457	0,5357	0,5713	**0,5802**	0,5590	0,5553	0,5908	**0,6036**
desvio padrão	0,0164	0,0023	0,0275	0,0112	0,0173	0,0246	0,0127	0,0172
LSTM	0,5106	0,4845	0,5203	**0,5317**	0,5159	0,4877	0,5213	**0,5360**
desvio padrão	0,0175	0,0638	0,0286	0,0124	0,0141	0,0183	0,0240	0,0391

Tabela 9 – Resultados da métrica ROC AUC para janela temporal de 5 dias.

Janela de 5 dias								
	Vetor de entrada = 150				Vetor de entrada = 300			
Algoritmo	Texto	Imagem	AOM	MASK	Texto	Imagem	AOM	MASK
HBOS	0,4176	0,4051	0,4370	**0,4849**	0,4003	0,3833	0,4398	**0,4587**
desvio padrão	0,0114	0,0190	0,0123	0,0194	0,0121	0,0020	0,0245	0,0260
kNN	0,3379	0,3151	0,3899	**0,4294**	0,3189	0,3078	0,3856	**0,4034**
desvio padrão	0,0153	0,0142	0,0164	0,0189	0,0293	0,0218	0,0096	0,0187
IForest	0,3036	0,2899	0,3146	**0,3394**	0,1957	0,1931	0,1994	**0,2276**
desvio padrão	0,0101	0,0203	0,0152	0,0091	0,0056	0,0122	0,0292	0,0281
Featuring Bagging	0,5349	0,5272	0,5496	**0,5648**	0,5123	0,4994	0,5138	**0,5191**
desvio padrão	0,0183	0,0213	0,0257	0,0204	0,0284	0,0228	0,0217	0,0215
Autoencoder	0,5472	0,5397	0,5769	**0,6149**	0,5618	0,5594	0,6012	**0,6154**
desvio padrão	0,0273	0,0196	0,0074	0,0170	0,0036	0,0274	0,0150	0,0180
LSTM	0,5171	0,4879	0,5268	**0,5395**	0,5141	0,4917	0,5359	**0,5445**
desvio padrão	0,0420	0,0294	0,0132	0,0148	0,0131	0,0343	0,0158	0,0174

Tabela 10 – Resultados da métrica ROC AUC para janela temporal de 10 dias.

Janela de 10 dias								
	Vetor de entrada = 150				Vetor de entrada = 300			
Algoritmo	Texto	Imagem	AOM	MASK	Texto	Imagem	AOM	MASK
HBOS	0,6160	0,5850	0,6191	**0,6313**	0,5957	0,5811	0,6104	**0,6229**
desvio padrão	0,0052	0,0214	0,0254	0,0121	0,0151	0,0153	0,0251	0,0160
kNN	0,5186	0,5102	0,5272	**0,5576**	0,5066	0,4911	0,5216	**0,5322**
desvio padrão	0,0265	0,0188	0,0121	0,0198	0,0041	0,0154	0,0397	0,0211
IForest	0,3843	0,3621	0,3880	**0,4869**	0,2105	0,2040	0,2168	**0,4151**
desvio padrão	0,0118	0,0291	0,0234	0,0264	0,0178	0,0046	0,0307	0,0199
Featuring Bagging	0,6417	0,5765	0,6585	**0,6905**	0,6410	0,5894	0,6442	**0,6699**
desvio padrão	0,0126	0,0286	0,0395	0,0114	0,0287	0,0039	0,0376	0,0094
Autoencoder	0,6832	0,6749	0,7055	**0,7353**	0,7003	0,6786	0,7118	**0,7768**
desvio padrão	0,0260	0,0228	0,0243	0,0325	0,0199	0,0149	0,0389	0,0196
LSTM	0,6052	0,5975	0,6150	**0,6292**	0,6121	0,6014	0,6241	**0,6390**
desvio padrão	0,0227	0,0109	0,0163	0,0163	0,0146	0,0162	0,0238	0,0153

Tabela 11 – Resultados da métrica ROC AUC para janela temporal de 20 dias.

Janela de 20 dias								
	Vetor de entrada = 150				Vetor de entrada = 300			
Algoritmo	Texto	Imagem	AOM	MASK	Texto	Imagem	AOM	MASK
HBOS	0,4651	0,4461	0,4704	**0,5198**	0,4586	0,4355	0,4693	**0,5083**
desvio padrão	0,0130	0,0187	0,0263	0,0041	0,0160	0,0293	0,0072	0,0262
kNN	0,3592	0,3357	0,4053	**0,4768**	0,3276	0,3132	0,3946	**0,4590**
desvio padrão	0,0070	0,0043	0,0091	0,0173	0,0286	0,0194	0,0273	0,0081
IForest	0,3036	0,2923	0,3269	**0,3552**	0,1957	0,1931	0,1994	**0,2276**
desvio padrão	0,0204	0,0299	0,0289	0,0228	0,0141	0,0140	0,0129	0,0155
Featuring Bagging	0,5404	0,5315	0,5531	**0,5863**	0,5128	0,5102	0,5331	**0,5409**
desvio padrão	0,0099	0,0049	0,0127	0,0148	0,0086	0,0024	0,0218	0,0134
Autoencoder	0,5578	0,5426	0,5903	**0,6410**	0,5826	0,5594	0,6012	**0,6512**
desvio padrão	0,0279	0,0175	0,0083	0,0128	0,0092	0,0189	0,0074	0,0230
LSTM	0,5329	0,5235	0,5451	**0,5705**	0,5316	0,5367	0,5492	**0,5811**
desvio padrão	0,0116	0,0165	0,0197	0,0185	0,2128	0,0113	0,0176	0,0117

Nos experimentos, a abordagem da fusão proposta usando MASK R-CNN treinada com a base de dados COCO mostraram melhores resultados do que a fusão realizada com AOM. Isso foi observado para todos os algoritmos não supervisionados aplicados nos experimentos. O melhor resultado foi obtido pelo algoritmo de *Autoencoder* com 0,6036 ROC AUC para janela de 1 dia, 0,6154 ROC AUC para janela de 5 dias, 0,7768 ROC AUC para janela de tamanho de 10 dias e 0,6512 ROC AUC para janela de tamanho de 20 dias.

O aumento progressivo do ROC AUC para diferentes janelas é atribuído ao desbalanceamento dos dados. A hipótese é que as janelas maiores possuem mais amostras de treinamento e validação que nas janelas menores. Além disso, o tamanho do vetor de entrada e o tipo de algoritmo não supervisionado impacta no resultado da identificação das novidades, como reportado nesta e em outras pesquisas (AGGARWAL, 2016), (AGGARWAL; SATHE, 2015) (AGGARWAL; SUBBIAN, 2012) (ROSA et al., 2018). Também houve uma queda no ROC AUC das janelas 10 e 20 dias, e isso se deve a uma redução na taxa dos falsos positivos. São *tweets* que são reportados novidade, mas, na realidade, não são novidades. Uma possível hipótese é com o aumento da janela, aumenta não apenas a quantidade de *tweets*, mas também aumenta a quantidade de *tweets* com traços de impureza que podem confundir o algoritmo de classificação, como por exemplo: *tweets* com imagens diferente do significado textual.

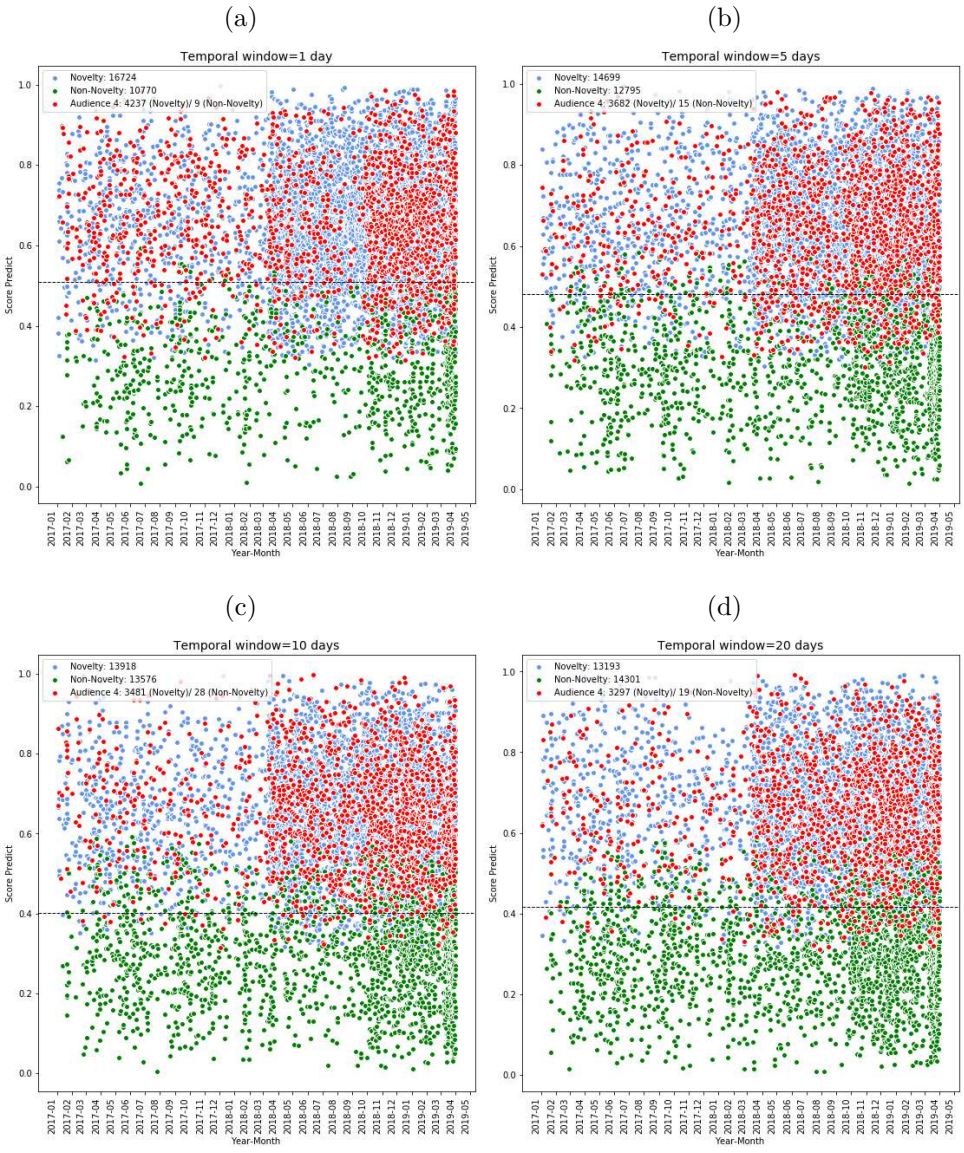
O método AOM realiza a fusão das probabilidades combinando a função de média e uma variante baseada em *rank* da função de maximização, buscando extrair o melhor das duas técnicas. A fusão que usa apenas a média é particularmente desejável para pequenos conjuntos de dados, por causa da sua robustez, por outro lado, a função de maximização enfatiza diferentes *outliers* que estão ocultos na função de média (AGGARWAL; SATHE, 2015). Entretanto, a fusão AOM não se mostrou satisfatória neste experimento, porque embora exista um pequeno conjunto de dados em cada janela, os dados são heterogêneos, desbalanceados e com muito ruído, impactando na fusão. Como a fusão usando a MASK R-CNN ocorre no *pipeline* de entrada, não tem o impacto de realizar a fusão de distintas probabilidades dos algoritmos na saída.

Outros resultados podem ser analisados nos gráficos da Figura 54 que ilustram os *tweets* ao longo do tempo e a pontuação deles obtida pelo algoritmo Autoencoder com a Fusão MASK R-CNN. Os *tweets* são marcados conforme os rótulos na base de dados: novidades, em azul, e não novidades, em verde. Em vermelho são marcados os *tweets* que têm audiência 4. Observa-se das legendas nos gráficos que grande parte dos *tweets* com audiência 4 são novidades, por exemplo, na janela temporal de 1 dia, 4237 são novidades contra 9 que não são. O eixo x indica o tempo de criação de cada *tweet* (em meses e anos) e o eixo y a pontuação (ou erro de reconstrução, ver Seção 3.10) gerada pelo algoritmo. Quanto maior for a pontuação, maior a chance da amostra ser uma novidade. A linha

tracejada nos gráficos indica o limiar médio normalizado (Min-Máx (NICOLAS, 2015)) usado sobre a pontuação para classificar o que é novidade (acima da linha) e o que não é. O limiar é definido na fase de treino e validação, tal que seu valor é selecionado buscando maximizar a ROC AUC no treino e validação.

Nota-se na Figura 54 que a grande maioria dos *tweets* com audiência 4 estão acima do limiar médio, e portanto muitos são identificados como novidades pelo modelo proposto. Isso indica que os *tweets* que aparentam ser de maior relevância em relação a ser novidade, são identificados pelo algoritmo proposto neste trabalho. Além disso, grande parte das não novidades se encontram abaixo do limiar de pontuação.

Figura 54 – Resultado para janelas temporais de 1, 5, 10 e 20 dias.

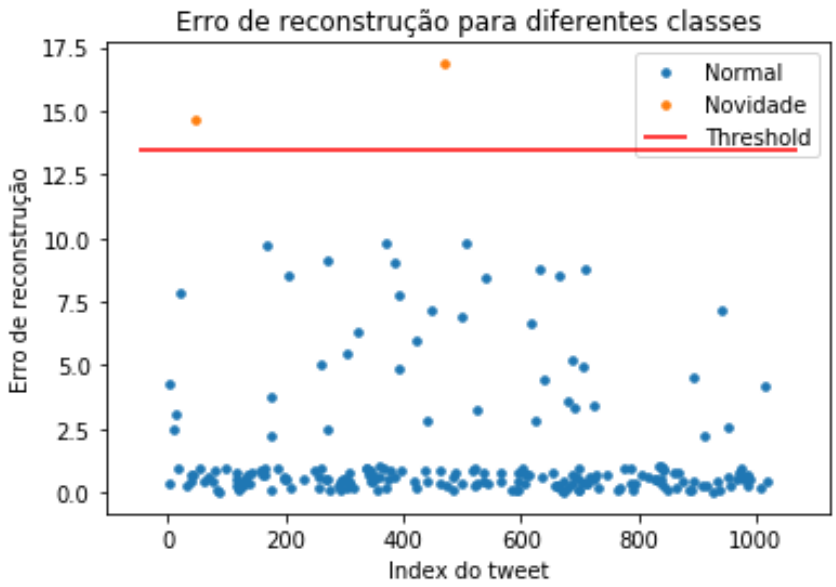


Autoria própria.

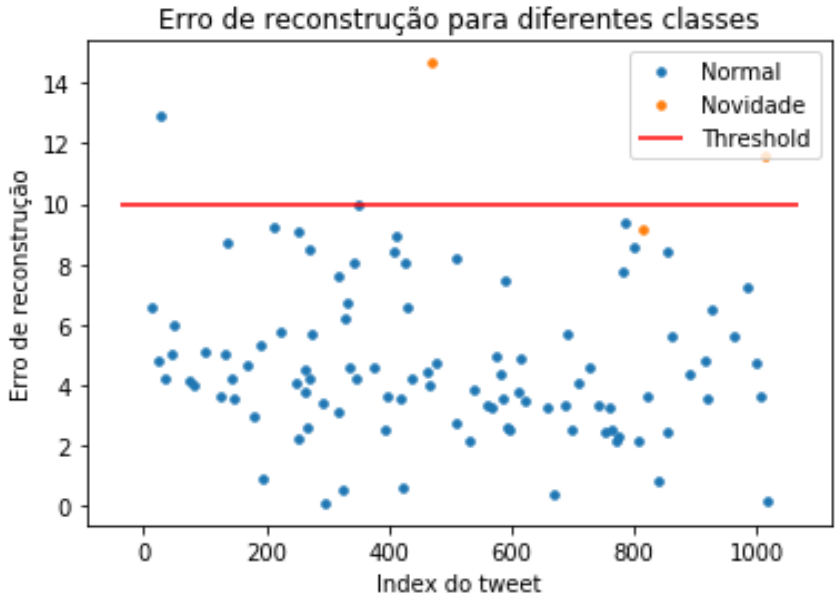
Os gráficos (a) e (b) da Figura 55 são importantes na análise da rede *Autoencoder*, pois apresenta a classificação de duas fatias temporais distintas da janela de 10 dias. Os pontos em laranja e azul representam as novidades e não novidades, respectivamente. A linha vermelha representa o limiar. Como pode ser visto, o algoritmo tem seu limiar dinâmico conforme a pontuação gerada pelo algoritmo e detalhado na Seção 3.10. A Figura 55 mostra que o algoritmo *Autoencoder* consegue uma boa separabilidade, por meio de um limiar, entre as novidades e não novidades de duas distintas fatias temporais.

Figura 55 – Gráfico representando o erro de reconstrução de janelas temporais distintas.

(a)



(b)



Autoria própria.

Segundo estatísticas, a proporção de amostras não estruturados em formato de texto é maior que imagens nas mídias sociais (ZEPHORIA, 2019). Portanto, foi analisado o quanto que a variação de proporção de *tweets* contendo somente texto, entre o total de *tweets*, pode afetar na detecção de novidades. Assim, foi realizado um experimento, para janelas temporais de 10 dias, variando a proporção de *tweets* com somente texto (as imagens foram desconsideradas nestes casos). Por exemplo, 40/60% significa que 40% das amostras incluem texto e imagem, e 60% apenas texto. A Tabela 12 mostra o resultado do experimento para 4 diferentes proporções, variando de 10% a 40% de *tweets* com texto e imagem. Para cada proporção de amostras, foram selecionados de forma aleatória os *tweets* com somente texto e aqueles com texto e imagem. Este experimento mostrou que, quando a proporção de imagens é maior ou igual a 30%, imagens adicionam informações relevantes ao texto as quais ajudam na melhoria da qualidade de classificação das novidades. Por exemplo, quando utilizado somente informação textual, nas mesmas condições do experimento (Tabela 12), a ROC AUC é de 0,7003. Gao et al. (2015) indicam que 30% dos blogs publicam conteúdo com imagens, mostrando, portanto, que o método proposto é relevante.

Tabela 12 – Resultados com diferentes proporções de texto e imagens usando a métrica ROC AUC para a janela temporal de 10 dias.

Janela 10 dias - Autoencoder		
	Vetor de entrada = 300	
Algoritmo	MASK	AOM
40/60%	**0.7615**	0.7549
desvio padrão	0.0115	0.0239
30/70%	**0.7327**	0.7110
desvio padrão	0.0365	0.0271
20/80%	**0.7027**	0.7005
desvio padrão	0.0155	0.0272
10/90%	**0.6982**	0.6954
desvio padrão	0.0118	0.0162

A Tabela 13 apresenta o tempo médio de treino e validação do algoritmo mais lento: LSTM.

Tabela 13 – Tempo de treino e validação do algoritmo LSTM.

Janela temporal	Tempo médio
1 dia	58 segundos
5 dias	1 minuto e 8 segundos
10 dias	2 minutos e 22 segundos
20 dias	4 minutos e 15 segundos

Como pode ser observado, o tempo de treino e validação varia de acordo com o tamanho de configuração da janela, porém são relativamente curtos em relação ao intervalo de tempo considerado.

5.5 Discussões

A fusão MASK R-CNN identifica rótulos na imagem, que adicionam novas informações que podem melhorar a classificação. Por exemplo, analisando o texto que veio no *tweet* junto com a imagem mostrada na Figura 56(a) *"10-year-old labrador and all-round good boy fred has become dad to nine ducklings"* e o texto associado à Figura 56(b) *"illegal puppy farming needs to stop"*, o primeiro texto menciona uma espécie de Labrador e o segundo texto se refere ao animal mais jovem, apesar de estarem falando de assuntos similares, as terminologias usadas são diferentes, tornando a classificação mais difícil. A rede MASK R-CNN pode reconhecer o rótulo *"dog"* na cena com alta confiança, e assim adicionar informação comum aos dois *tweets*, trazendo mais assertividade ao algoritmo de aprendizado. Portanto, uma imagem pode carregar informação, ainda que a qualidade da imagem não seja boa, e que pode complementar uma informação ausente no texto do *tweet*.

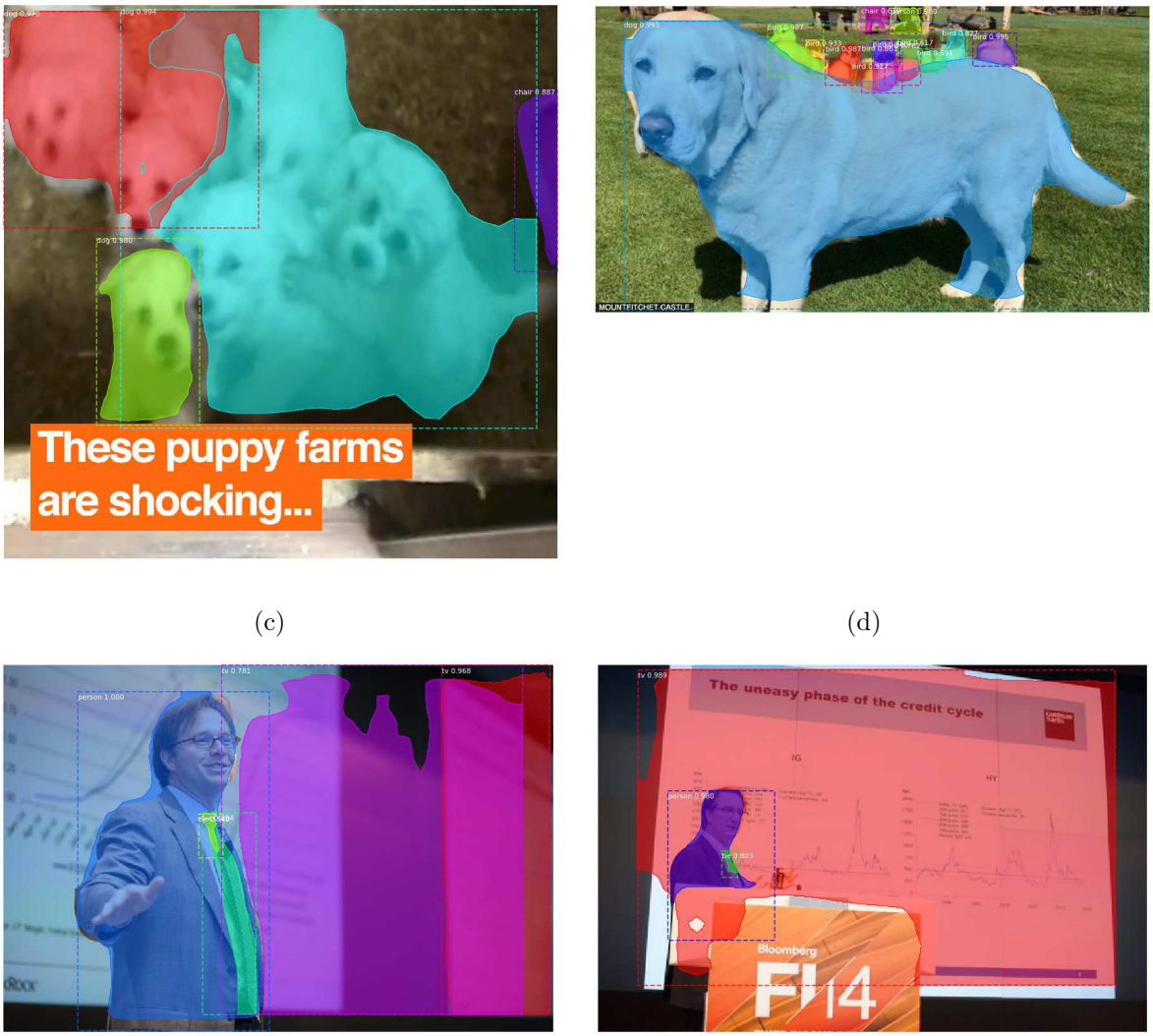
Em outro exemplo, as Figuras 56(c) e 56(d) têm similaridades, enquanto os textos da Fig. 56(c) é *"how I stopped worrying and learned to love the link : @blackrock jeff rosenberg at fi14 in ny #bbfi"* e a Fig. 56(d) é: *"head of global credit strategy @goldmansachs discusses credit market trends in event fi14 in ny #bbgfi"*. Neste caso, os textos contêm semânticas distintas, embora compartilhem apenas uma única *hashtag*, os *tweets* referem-se ao mesmo evento. *Hashtags* em comum não trazem ganho textual, uma vez que a rede pré-treinada do Google *embeddings*, usada neste trabalho, não reconhece a palavra com o *hashtag* e retorna um vetor nulo. A rede MASK R-CNN pode reconhecer os rótulos *"tv"*, *"person"* e *"tie"* na cena. Portanto, a adição de palavras-chave ao *tweet* aumenta a chance de uma correta classificação.

Outra questão importante na análise é que, a ROC AUC diminui quando o vetor de entrada aumenta a dimensão de 150 para 300, exceto o algoritmo *Autoencoder*. A literatura destaca que algoritmos baseados em distância têm desvantagem com dados de alta dimensão (AGGARWAL, 2016). Outra razão é que nenhum dos algoritmos não supervisionados usados tem a capacidade de selecionar/extrair características ou de serem formados por *ensembles*, exceto os algoritmos *FBagging* e *Autoencoder*.

Apesar do algoritmo LSTM se comportar como um extrator de características, os experimentos não o apontam como um algoritmo eficaz. A literatura relata que redes LSTM têm resultados satisfatórios em aplicações que usam dados sequenciais uniformemente amostrados (SAHIN; KOZAT, 2018), ou seja, os dados são amostrados em uma frequência fixa de amostragem. Entretanto, os *tweets* postados em uma janela temporal são não-uniformemente amostrados, ocasionando um perda na eficácia do algoritmo. Uma outra hipótese, é que a versão não supervisionada pode perder em acurácia para a versão supervisionada, isso já foi evidenciado na literatura por Jarrett et al. (2009) e Collingwood e Wilkerson (2012) em outros algoritmos.

Figura 56 – Exemplos de *tweets* com imagens nas quais usam a fusão com informações agregadas da base de dados COCO. Os seguintes objetos nas cenas são: (a) *dogs* (0,970); (0,994); (0,980) e *chair* (0,887); (b) *dog* (0,993); *birds* (0,987); (0,933); (0,917), *chair* (0,610) e *person* (0,500); (c) *person* (1,000), *tv* (0,968) e *tie* (0,540); (d) *person* (0,980), *tv* (0,989) e *tie* (0,803).

(a) (b)



(c) (d)

Autoria própria.

A proposta de qualificação da novidade tem um papel importante, pois permite filtragem dos *tweets* que se destacam pela audiência. Portanto, a aplicação do filtro influenciou a eliminação das novidades "ruidosas", como *tweets* com propaganda: *"strengthening and enriching our communities is one of the core principles at Bloomberg:"*. Outros *tweets* foram destacados com o filtro de audiência, por exemplo, recorde no mercado de ações: *"rt @business: breaking: the dow hits 20,000 for the first time"*. O primeiro *tweet* exemplificando uma propaganda é de fato uma novidade uma vez que se enquadra na definição proposta por este trabalho, e não se pode restringir a novidade pelo assunto.

A Tabela 14 mostra exemplos de *tweets*, e o filtro de audiência 4 destaca os mais importantes eventos do dia. A primeira coluna da tabela refere-se a audiência. A segunda coluna da tabela refere-se ao rótulo: *Novidade* (N) ou *Não Novidade* (NN). A terceira coluna da tabela refere-se ao tempo de postagem do *tweet* e a última coluna é o texto do *tweet*. A Fig. 57 mostra a novidade, com audiência 4, postada 2018-06-12 01:08:35.

Tabela 14 – Uma amostra do conjunto de dados de *tweets* postados no dia 2018-06-12.

A	R	Tempo	Texto
4	N	01:00:34	@nbcnews kim jong un has arrived for the summit in singapore.person car bird chair
3	NN	01:00:34	president trump has arrived for the summit in singapore.person tie car
4	N	01:08:35	In pictures: President Trump meets Kim Jong Un – CNN.com
4	NN	01:10:22	A historic handshake ... but what did the Trump-Kim summit really achieve?
3	NN	01:17:28	when trump met kim. clearer photos of that moment in history
3	NN	01:18:21	Trump and Kim finally meet
3	NN	02:34:08	watch: bilateral meetings begin between kim jong un
3	NN	03:40:54	watch: kim jong un, trump shake hands for the first timeperson kite
1	N	04:14:08	watch: thirsty koala enjoys a drinkperson dog
2	N	15:21:57	distracted deputy comes disturbingly close to slamming into student boarding buschair person

Figura 57 – *Tweet* postado em 2018-06-12 01:08:35.

Autoria (CNN, 2018).

Uma outra característica da janela temporal é a mudança no rótulo de novidade. Isso implica na mudança na distribuição fundamental do problema, conhecido na literatura como *concept drift*. Um *tweet* de exemplo é: *"rt @theterminal: market close today. #Dow20k"*, que na janela de 10 dias é considerado novidade por causa dos *tweets* nas janelas temporais de presente e passado. Entretanto, não é novidade na janela de 20 dias, porque a agência

de notícias *Bloomberg* reportou primeiro com o *tweet*: *"rt @business: breaking: the dow hits 20.000 for the first time"*. Esse exemplo é interessante, pois trata-se de identificar notícias em primeira mão, conhecidas como notícias de *breaking news*.

6 CONCLUSÕES FINAIS

Neste capítulo é realizada uma síntese das conclusões do trabalho e uma descrição de ações propostas para a continuidade desta pesquisa.

6.1 Compêndio

Este trabalho propõe detectar novidades em mídias sociais usando fusão de dados heterogêneos. Para este fim, foram apresentados dois métodos de fusão: a fusão do texto e da imagem usando a rede MASK R-CNN, treinada com a base de dados COCO, e a fusão na saída dos algoritmos não supervisionados com a técnica AOM. O método de fusão usando MASK R-CNN mostrou bons resultados, superando a fusão AOM, e o uso de somente texto ou imagem na detecção de novidades.

Todos os algoritmos obtiveram melhores resultados usando apenas texto do que apenas imagem para classificação. Os desafios enfrentados em identificar uma novidade usando apenas imagens são maiores quando se trata de rede social, pois as imagens possuem baixa qualidade e em grande parte expressam significado amplo. Vale destacar que a fusão de dados produz melhores resultados dentre todas as técnicas comparadas. Foi observado experimentalmente que para a fusão de dados ter desempenho relevante na detecção de novidades, em relação a usar somente textos, a proporção de *tweets* com imagens e textos no conjunto de dados deve ser maior ou igual a 30%. Gao et al. (2015) informam que 30% dos blogs publicam conteúdo com imagens, indicando, portanto, que o método proposto é relevante.

Os algoritmos não supervisionados que apresentaram os melhores resultados foram: *Autoencoder* e o *Featuring Bagging*. Diferentemente de todos os algoritmos comparados, as redes LSTM fornecem ganhos com seus mecanismos de portões, que controlam a quantidade de informações que entram na rede e as informações do passado que são armazenadas em memória. Entretanto, as redes LSTM não apresentaram bons resultados neste trabalho. Embora apresentem bons resultados em aplicações usando dados com frequência fixa de amostragem, geralmente apresentam resultados ruins quando não estão uniformemente amostrados (SAHIN; KOZAT, 2018), como é o caso dos dados provenientes das redes sociais. Não há uma quantidade de *tweets* uniformes em cada intervalo temporal. Para contornar esta questão, é possível converter as amostras não uniformes em amostras uniformes empregando diversas técnicas, entretanto, como é relatado na literatura ainda não se consegue alcançar bons resultados com essas abordagens (SAHIN; KOZAT, 2018).

Outra questão importante para se alcançar bons resultados com os algoritmos de

aprendizado é definir adequadamente o tamanho do vetor de entrada. O tamanho do vetor de entrada é imprescindível para uma boa representatividade dos dados como já notado em outras pesquisas. O vetor com 150 dimensões produziu, em geral, melhores resultados do que o de 300 dimensões, no entanto, não houve unanimidade dos algoritmos, pois em alguns o resultado melhorava com o aumento da dimensão. Possivelmente a limitação da quantidade de dados para treinamento pode ter impacto no resultado.

A definição de novidade envolvendo janelas temporais (definida na Equação 4.1) e qualidade da novidade (definida na Equação 4.3) mostraram ser uma ferramenta de análise útil para detectar *tweets* livres de contextos específicos. Como resultado, a definição de novidade permitiu filtrar *tweets* com alta audiência, assim eliminando propaganda e outros *tweets* irrelevantes. Também é notado que quanto menor a janela temporal mais difícil é a classificação, sendo isso atribuído ao desbalanceamento das classes e a menor quantidade de amostras na fase de treino e validação.

Além disso, foi observado que pequenas janelas temporais tendem a identificar muitas novidades, enquanto grandes janelas temporais tendem a localizar poucas novidades. Em outras palavras, uma mudança no tamanho da janela temporal muda o percentual de exemplos rotulados como novidades e, portanto, implica em uma mudança na distribuição fundamental do problema, fenômeno este conhecido na literatura como *concept drift* (WANG; MINKU; YAO, 2018).

A mudança na distribuição impacta em dois fatores: a) na escolha dos algoritmos de detecção de novidade, porque, neste caso específico, métodos não paramétricos se ajustam melhor ao problema (AGGARWAL, 2016) (WANG; MINKU; YAO, 2018); b) a identificação apropriada do rótulo, uma vez que depende do tamanho da janela temporal (FAN, 2004) (FORMAN, 2006) (WAGNER et al., 2018).

6.2 Trabalhos futuros

O propulsor da continuidade deste trabalho de doutorado está nos resultados promissores, no aprofundamento da nova abordagem de fusão de dados heterogêneos realizados na entrada dos algoritmos, no fornecimento de uma nova base de dados com características singulares e na utilidade ferramental, que além de classificar as novidades, também a qualifica. É necessário ainda uma investigação mais profunda e aprimorada da pesquisa. Neste sentido, quatro novas ações são delineadas para a continuação desta pesquisa:

- **Ampliar a base de dados**: A ampliação de novos exemplos provenientes de distintas redes sociais possibilitará uma base de dados que permite realizar análises mais complexas e completas.

- **Realizar a fusão com outros extratores de características de imagem:** Explorar outros extratores de características de imagem (QIAN et al., 2018) (SARWAR et al., 2019) é uma forma de aprofundar e validar a fusão na entrada dos algoritmos de aprendizado não supervisionado.

- **Codificar o texto com técnicas de *embedding* bidirecionais:** Os novos *embeddings* do estado-da-arte, como o BERT da Google (DEVLIN et al., 2018) e o MT-DNN da Microsoft (LIU et al., 2019) não só codificam o texto em vetores, mas aprendem o contexto e a relação entre as palavras usando treinamento bidirecional, essa técnica é denominada *Masked LM*. As redes neurais bidirecionais são mais complexas e, portanto, conseguem capturar melhor as informações presentes numa sentença para fazer uma previsão melhor. Por exemplo: *Teddy bears are beautiful toys* e *Teddy Roosevelt, the president of United State*. Nessas duas frases, não é possível saber o que virá exatamente depois de *Teddy*, se apenas usar o contexto do lado esquerdo, como acontece nas LSTMs comuns. Também é necessário o contexto do lado direito para fazer uma previsão adequada sobre se a próxima palavra é *bears* ou *Roosevelt*. O contexto do lado direito é *toys* ou *president*. Assim, em *embeddings* bidirecionais sentenças de significados semelhantes teriam incorporações vetoriais semelhantes (DAULET, 2020).

  A intenção é verificar se os codificadores bidirecionais de textos por si só melhoram a detecção de novidades ou se a fusão proposta é necessária para aprimorar a detecção de novidades.

- **Comparar a arquitetura**: A comparação da arquitetura proposta neste trabalho com redes profundas geradoras de texto que descrevem sucintamente uma imagem, conhecidas como *Image Caption Generation* (AMRITKAR; JABADE, 2018).

Referências

AMRITKAR, C. and Jabade, V. Citado na página 59.

AGGARWAL, C.; REDDY, C. *Data Clustering: Algorithms and Applications*. CRC Press, 2013. (Chapman & Hall/CRC Data Mining and Knowledge Discovery Series). ISBN 9781466558229. Disponível em: <https://books.google.com.br/books?id=JU_SBQAAQBAJ>. Citado na página 28.

AGGARWAL, C. C. *Outlier Analysis*. 2nd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2016. ISBN 3319475770, 9783319475776. Citado 8 vezes nas páginas 20, 55, 56, 57, 93, 106, 110 e 115.

AGGARWAL, C. C.; SATHE, S. Theoretical foundations and algorithms for outlier ensembles. *SIGKDD Explor. Newsl.*, ACM, New York, NY, USA, v. 17, n. 1, p. 24–47, set. 2015. ISSN 1931-0145. Citado 4 vezes nas páginas 44, 89, 104 e 106.

AGGARWAL, C. C.; SATHE, S. *Outlier ensembles: An introduction*. [S.l.]: Springer, 2017. Citado 3 vezes nas páginas 38, 43 e 44.

AGGARWAL, C. C.; SUBBIAN, K. Event detection in social streams. In: *SDM*. SIAM / Omnipress, 2012. p. 624–635. ISBN 978-1-61197-282-5. Disponível em: <http://dblp.uni-trier.de/db/conf/sdm/sdm2012.html#AggarwalS12>. Citado 4 vezes nas páginas 21, 25, 35 e 106.

AGGARWAL, C. C.; YU, P. S. Outlier detection for high dimensional data. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 30, n. 2, p. 37–46, maio 2001. ISSN 0163-5808. Disponível em: <http://doi.acm.org/10.1145/376284.375668>. Citado na página 59.

AIELLO, L. M. et al. Sensing trending topics in twitter. *IEEE Trans. Multimedia*, v. 15, n. 6, p. 1268–1282, 2013. Disponível em: <https://doi.org/10.1109/TMM.2013.2265080>. Citado 3 vezes nas páginas 27, 35 e 80.

ALKHODAIR, S. A. et al. Detecting breaking news rumors of emerging topics in social media. *Information Processing & Management*, Elsevier, 2019. Citado 4 vezes nas páginas 8, 33, 34 e 35.

ALLAN, J. et al. Topic detection and tracking pilot study: Final report. In: *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*. Lansdowne, VA, USA: [s.n.], 1998. p. 194–218. 007. Citado na página 37.

ALOYSIUS, N.; GEETHA, M. A review on deep convolutional neural networks. *2017 International Conference on Communication and Signal Processing (ICCSP)*, p. 0588–0592, 2017. Citado 2 vezes nas páginas 44 e 47.

ALQHTANI, S. M.; LUO, S.; REGAN, B. Fusing text and image for event detection in twitter. *arXiv preprint arXiv:1503.03920*, 2015. Citado 2 vezes nas páginas 28 e 35.

AMORIM, M. et al. Analysing rotation-invariance of a log-polar transformation in convolutional neural networks. In: *2018 International Joint Conference on Neural*

Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018. [s.n.], 2018. p. 1–6. Disponível em: <https://doi.org/10.1109/IJCNN.2018.8489295>. Citado na página 24.

AMORIM, M. et al. Novelty detection in social media by fusing text and image into a single structure. *IEEE Access*, v. 7, p. 132786–132802, 2019. Citado na página 23.

AMORIM, M.; CIARELLI, P. M. Fusão de redes convolucionais heterogêneas para detecção de novidades em redes sociais. In: *2018 Congresso Brasileiro de Automátic, CBA 2018, João Pessoa, Brazil, September 9-12, 2018*. [S.l.: s.n.], 2018. Citado na página 24.

AMRITKAR, C.; JABADE, V. Image caption generation using deep learning technique. In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. [S.l.: s.n.], 2018. p. 1–4. Citado na página 116.

AN, J.; CHO, S. Variational autoencoder based anomaly detection using reconstruction probability. In: . [S.l.: s.n.], 2015. Citado na página 65.

ANALYTICSVIDHYA.COM. *Computer Vision Tutorial: Implementing Mask R-CNN for Image Segmentation (with Python Code)*. 2020. <http://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/>. [Online; accessed 12-Feb-2020]. Citado na página 52.

ANGIULLI, F.; PIZZUTI, C. Fast outlier detection in high dimensional spaces. In: *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*. London, UK, UK: Springer-Verlag, 2002. (PKDD '02), p. 15–26. ISBN 3-540-44037-2. Citado 2 vezes nas páginas 56 e 100.

ASMITA, S.; SHUKLA, K. Review on the architecture, algorithm and fusion strategies in ensemble learning. *International Journal of Computer Applications*, Citeseer, v. 108, n. 8, 2014. Citado 2 vezes nas páginas 43 e 47.

ATEFEH, F.; KHREICH, W. A survey of techniques for event detection in twitter. *Comput. Intell.*, Blackwell Publishers, Inc., Cambridge, MA, USA, v. 31, n. 1, p. 132–164, fev. 2015. ISSN 0824-7935. Disponível em: <http://dx.doi.org/10.1111/coin.12017>. Citado 2 vezes nas páginas 19 e 37.

AUDEBERT, N.; SAUX, B. L.; LEFÈVRE, S. Fusion of heterogeneous data in convolutional networks for urban semantic labeling. *2017 Joint Urban Remote Sensing Event (JURSE)*, p. 1–4, 2017. Citado na página 44.

BARNETT, V.; LEWIS, T. *Outliers in Statistical Data*. 2nd edition. ed. [S.l.]: Wiley, 1994. v. 37. (Wiley Series in Probability and Statistics, 2). ISBN 0471930946. Citado na página 19.

BASU, S.; BILENKO, M.; MOONEY, R. J. A probabilistic framework for semi-supervised clustering. In: ACM. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2004. p. 59–68. Citado na página 37.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, IEEE Computer Society, Washington, DC, USA, v. 35, n. 8, p. 1798–1828, ago. 2013. ISSN 0162-8828. Citado na página 84.

BEYER, K. S. et al. When is "nearest neighbor" meaningful? In: *Proceedings of the 7th International Conference on Database Theory*. London, UK, UK: Springer-Verlag, 1999. (ICDT '99), p. 217–235. ISBN 3-540-65452-6. Disponível em: <http://dl.acm.org/citation.cfm?id=645503.656271>. Citado na página 59.

BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738. Citado na página 53.

BRAGA, A. C. da S. *CURVAS ROC: ASPECTOS FUNCIONAIS E APLICAÇÕES*. Tese (Doutorado) — Universidade do Minho, Portugal, 12 2000. Citado na página 93.

BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996. Citado na página 43.

BREIMAN, L. Bagging predictors. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 24, n. 2, p. 123–140, ago. 1996. ISSN 0885-6125. Disponível em: <http://dx.doi.org/10.1023/A:1018054314350>. Citado na página 59.

BREIMAN, L. *Bias, variance, and arcing classifiers*. [S.l.], 1999. Citado na página 39.

BREUNIG, M. M. et al. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 29, n. 2, p. 93–104, maio 2000. ISSN 0163-5808. Disponível em: <http://doi.acm.org/10.1145/335191.335388>. Citado na página 59.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 41, n. 3, p. 15:1–15:58, jul. 2009. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/1541880.1541882>. Citado na página 20.

CHEN, J. et al. Outlier detection with autoencoder ensembles. In: . [S.l.: s.n.], 2017. p. 90–98. ISBN 978-1-61197-497-3. Citado 4 vezes nas páginas 56, 65, 66 e 100.

CLIFTON, L. et al. Identification of patient deterioration in vital-sign data using one-class support vector machines. In: . [S.l.: s.n.], 2011. p. 125–131. Citado na página 37.

CNN. *CNN NEWS*. 2018. Disponível em: <https://edition.cnn.com/2018/06/12/opinions/trump-kim-us-north-korea-summit-reactions-opinion/index.html>. Citado na página 112.

COLLINGWOOD, L.; WILKERSON, J. Tradeoffs in accuracy and efficiency in supervised learning methods. *Journal of Information Technology & Politics*, Taylor & Francis, v. 9, n. 3, p. 298–318, 2012. Citado na página 110.

DAULET, N. *BERT Model Embeddings are not as good as you think*. 2020. Disponível em: <https://towardsdatascience.com/cutting-edge-bert-nlp-model-bb0bfc8b7aec>. Citado na página 116.

DAVIDSON, T. et al. Automated hate speech detection and the problem of offensive language. *CoRR*, abs/1703.04009, 2017. Disponível em: <http://arxiv.org/abs/1703.04009>. Citado na página 95.

deeplearningbook. *Arquitetura de Redes Neurais Long Short Term Memory*. 2019. Disponível em: <deeplearningbook.com.br/arquitetura-de-redes-neurais-long-short-term-memory/>. Citado na página 67.

DEVLIN, J. et al. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. Disponível em: <http://arxiv.org/abs/1810.04805>. Citado 2 vezes nas páginas 73 e 116.

DIEHL, C. P.; HAMPSHIRE, J. B. Real-time object classification and novelty detection for collaborative video surveillance. In: IEEE. *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*. [S.l.], 2002. v. 3, p. 2620–2625. Citado na página 37.

DING, X. et al. Mining user consumption intention from social media using domain adaptive convolutional neural network. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 2015. (AAAI'15), p. 2389–2395. ISBN 0-262-51129-0. Citado 2 vezes nas páginas 27 e 35.

DODGE, S.; KARAM, L. Understanding how image quality affects deep neural networks. In: IEEE. *Quality of Multimedia Experience (QoMEX), 2016 Eighth International Conference on*. [S.l.], 2016. p. 1–6. Citado na página 21.

FAN, W. Systematic data selection to mine concept-drifting data streams. In: ACM. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2004. p. 128–137. Citado na página 115.

FAWCETT, T. *ROC Graphs: Notes and Practical Considerations for Researchers*. Palo Alto, 2004. Citado na página 93.

FORMAN, G. Tackling concept drift by temporal inductive transfer. In: ACM. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. [S.l.], 2006. p. 252–259. Citado na página 115.

FREUND, Y.; SCHAPIRE, R. E. Experiments with a new boosting algorithm. In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996. (ICML'96), p. 148–156. ISBN 1-55860-419-7. Disponível em: <http://dl.acm.org/citation.cfm?id=3091696.3091715>. Citado na página 59.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, v. 36, p. 193–202, 1980. Citado na página 44.

GAO, Y. et al. Multimedia social event detection in microblog. In: *MultiMedia Modeling*. Cham: Springer International Publishing, 2015. p. 269–281. ISBN 978-3-319-14445-0. Citado 3 vezes nas páginas 21, 109 e 114.

GHELANI, N. et al. Event detection on curated tweet streams. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2017. (SIGIR '17), p. 1325–1328. ISBN 978-1-4503-5022-8. Disponível em: <http://doi.acm.org/10.1145/3077136.3084141>. Citado na página 21.

GHOSH, A.; VEALE, T. Fracking sarcasm using neural network. In: *WASSA@NAACL-HLT*. [S.l.: s.n.], 2016. Citado na página 32.

GO, A.; BHAYANI, R.; HUANG, L. *Twitter sentiment classification using distant supervision*. 2009. Disponível em: <https://www.kaggle.com/kazanova/sentiment140>. Citado na página 95.

GOLDBARG, M.; GOLDBARG, E. *Grafos*. Elsevier Editora Ltda., 2012. ISBN 9788535257182. Disponível em: <https://books.google.com.br/books?id=_aoX3ZrMhCIC>. Citado na página 75.

GOLDBERG, Y. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, AI Access Foundation, v. 57, p. 345–420, Nov 2016. ISSN 1076-9757. Disponível em: <http://dx.doi.org/10.1613/jair.4992>. Citado na página 70.

GOLDSTEIN, M.; DENGEL, A. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. In: WöLFL, S. (Ed.). *KI-2012: Poster and Demo Track*. [S.l.]: Online, 2012. p. 59–63. Citado 5 vezes nas páginas 56, 57, 58, 59 e 100.

GONZALEZ, R.; WOODS, R. *Digital Image Processing*. Pearson, 2018. ISBN 9780133356724. Disponível em: <https://books.google.com.br/books?id=0F05vgAACAAJ>. Citado 2 vezes nas páginas 28 e 29.

GOODFELLOW, I. J.; YOSHUA; COURVILLE, A. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado 6 vezes nas páginas 45, 46, 47, 65, 67 e 83.

GUPTA, M. et al. *Outlier Detection for Temporal Data*. [S.l.]: Morgan & Claypool Publishers, 2014. ISBN 1627053751, 9781627053754. Citado na página 38.

HE, K. et al. Mask R-CNN. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. [s.n.], 2017. p. 2980–2988. Disponível em: <https://doi.org/10.1109/ICCV.2017.322>. Citado 4 vezes nas páginas 48, 49, 50 e 89.

HENDRICKSON, S. et al. *Trend Detection in Social Data*. 2019. <developer.twitter.com>. [Online; accessed 15-July-2019]. Citado na página 22.

HONG, L.; DAN, O.; DAVISON, B. D. Predicting popular messages in twitter. In: *Proceedings of the 20th International Conference Companion on World Wide Web*. New York, NY, USA: ACM, 2011. (WWW '11), p. 57–58. ISBN 978-1-4503-0637-9. Citado na página 22.

HUANG, A. Similarity measures for text document clustering. In: *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*. [S.l.: s.n.], 2008. v. 4, p. 9–56. Citado na página 81.

IBRAHIM, R. et al. Tools and approaches for topic detection from twitter streams: Survey. *Knowledge and Information Systems*, Springer-Verlag New York, Inc., New York, NY, USA, v. 54, n. 3, p. 511–539, mar. 2018. ISSN 0219-1377. Disponível em: <https://doi.org/10.1007/s10115-017-1081-x>. Citado na página 20.

IZZELDIN, H.; ASIRVADAM, V. S.; SAAD, N. Online sliding-window based for training mlp networks using advanced conjugate gradient. In: *2011 IEEE 7th International Colloquium on Signal Processing and its Applications*. [S.l.: s.n.], 2011. p. 112–116. Citado na página 96.

JAMES, G. et al. *An Introduction to Statistical Learning: With Applications in R*. [S.l.]: Springer Publishing Company, Incorporated, 2014. ISBN 1461471370, 9781461471370. Citado 4 vezes nas páginas 39, 41, 54 e 55.

JARRETT, K. et al. What is the best multi-stage architecture for object recognition? In: IEEE. *2009 IEEE 12th international conference on computer vision*. [S.l.], 2009. p. 2146–2153. Citado na página 110.

JOULIN, A. et al. Bag of tricks for efficient text classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Association for Computational Linguistics, 2017. Disponível em: <http://dx.doi.org/10.18653/v1/e17-2068>. Citado na página 73.

JURAFSKY, D. *Speech & language processing*. [S.l.]: Pearson Education India, 2000. Citado na página 85.

KANEKO, T.; YANAI, K. Event photo mining from twitter using keyword bursts and image clustering. *Neurocomput.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 172, n. C, p. 143–158, jan. 2016. ISSN 0925-2312. Citado 2 vezes nas páginas 29 e 35.

Karl, H. . *10 news stories that broke on Twitter first*. 2010. Disponível em: <https://www.techradar.com/news/world-of-tech/internet/10-news-stories-that-broke-on-twitter-first-719532>. Citado na página 79.

Karpath, A. *Convolutional Neural Networks for Visual Recognition*. 2019. Disponível em: <http://cs231n.github.io/convolutional-networks/>. Citado 3 vezes nas páginas 46, 48 e 49.

KENTER, T.; BORISOV, A.; RIJKE, M. de. Siamese cbow: Optimizing word embeddings for sentence representations. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2016. Disponível em: <http://dx.doi.org/10.18653/v1/P16-1089>. Citado na página 74.

KHATER, S.; GRACANIN, D.; ELMONGUI, H. Personalized recommendation for online social networks information: Personal preferences and location-based community trends. *IEEE Transactions on Computational Social Systems*, PP, p. 1–17, 07 2017. Citado na página 22.

KIM, Y. Convolutional neural networks for sentence classification. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. [S.l.: s.n.], 2014. p. 1746–1751. Citado 2 vezes nas páginas 31 e 32.

KNORR, E. M.; NG, R. T. Algorithms for mining distance-based outliers in large datasets. In: *Proceedings of the 24rd International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998. (VLDB '98), p. 392–403.

ISBN 1-55860-566-5. Disponível em: <http://dl.acm.org/citation.cfm?id=645924.671334>. Citado na página 59.

KOEHN, P. *Statistical Machine Translation*. 1st. ed. New York, NY, USA: Cambridge University Press, 2010. ISBN 0521874157, 9780521874151. Citado 4 vezes nas páginas 69, 70, 71 e 72.

KONG, E. B.; DIETTERICH, T. G. Error-correcting output coding corrects bias and variance. In: *Machine Learning Proceedings 1995*. [S.l.]: Elsevier, 1995. p. 313–321. Citado na página 43.

KONG, E. B.; DIETTERICH, T. G. Error-correcting output coding corrects bias and variance. In: *In Proceedings of the Twelfth International Conference on Machine Learning*. [S.l.]: Morgan Kaufmann, 1995. p. 313–321. Citado na página 59.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems 25*. [S.l.]: Curran Associates, Inc., 2012. p. 1097–1105. Citado na página 44.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105. Citado na página 66.

KUMAR, R. Analysis of shape alignment using euclidean and manhattan distance metrics. In: IEEE. *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)*. [S.l.], 2017. p. 326–331. Citado na página 81.

KWON, S.; CHA, M.; JUNG, K. *Rumor detection over varying time windows.* Harvard Dataverse, 2016. Disponível em: <https://doi.org/10.7910/DVN/BFGAVZ>. Citado na página 95.

LA TIMES NEWS. *Flickr reaches 6 billion photos uploaded*. 2018. Disponível em: <https://latimesblogs.latimes.com/technology/2011/08/flickr-reaches-6-billion-photos-uploaded.html>. Citado na página 21.

LANGFORD, E. *Quartiles in Elementary Statistics*. 2006. <http://jse.amstat.org/v14n3/langford.html>. [Online; accessed 15-July-2019]. Citado na página 79.

LAZAREVIC, A.; KUMAR, V. Feature bagging for outlier detection. In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. New York, NY, USA: ACM, 2005. (KDD '05), p. 157–166. ISBN 1-59593-135-X. Citado 4 vezes nas páginas 43, 56, 60 e 100.

LE, Q. V.; MIKOLOV, T. Distributed representations of sentences and documents. In: *ICML*. JMLR.org, 2014. (JMLR Workshop and Conference Proceedings, v. 32), p. 1188–1196. Disponível em: <http://dblp.uni-trier.de/db/conf/icml/icml2014.html#LeM14>. Citado na página 73.

LECUN, Y. et al. Handwritten digit recognition with a back-propagation network. In: TOURETZKY, D. S. (Ed.). *Advances in Neural Information Processing Systems 2*. [S.l.]: Morgan-Kaufmann, 1990. p. 396–404. Citado na página 44.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. *CoRR*, abs/1405.0312, 2014. Citado 3 vezes nas páginas 89, 91 e 92.

LISTER, M. *33 Mind Boggling Instagram Stats and Facts for 2018*. 2019. <https://www.wordstream.com/blog/ws/2017/04/20/instagramstatistics>. [Online; accessed 27-March-2020]. Citado na página 21.

LIU, F. T.; TING, K. M.; ZHOU, Z.-H. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, ACM, New York, NY, USA, v. 6, n. 1, p. 3:1–3:39, mar. 2012. ISSN 1556-4681. Citado 4 vezes nas páginas 56, 61, 63 e 100.

LIU, X. et al. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504, 2019. Disponível em: <http://arxiv.org/abs/1901.11504>. Citado na página 116.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. ISBN 978-0-521-86571-5. Disponível em: <http://nlp.stanford.edu/IR-book/information-retrieval-book.html>. Citado na página 25.

MAO, J. et al. Training and evaluating multimodal word embeddings with large-scale web annotated images. Curran Associates, Inc., p. 442–450, 2016. Citado 3 vezes nas páginas 30, 31 e 35.

MAO, J. et al. *Learning like a Child: Fast Novel Visual Concept Learning from Sentence Descriptions of Images*. 2015. Citado na página 30.

MASUD, M. A. et al. Cluster survival model of concept drift in load profile data. *IEEE Access*, v. 6, p. 51269–51285, 2018. Citado na página 22.

MCQUAIL, D. *Audience Analysis*. SAGE Publications, 1997. (Audience Analysis). ISBN 9780761910022. Disponível em: <https://books.google.com.br/books?id=InB0Ib_ktbQC>. Citado na página 78.

MEI, M. et al. Using semantic clustering and autoencoders for detecting novelty in corpora of short text. In: *IJCNN, 2018 International Joint Conference on Neural Networks*. [S.l.]: IEEE Explorer, 2018. p. 323–330. Citado 2 vezes nas páginas 31 e 35.

MELE, I.; BAHRAINIAN, S. A.; CRESTANI, F. Event mining and timeliness analysis from heterogeneous news streams. *Information Processing & Management*, Elsevier, v. 56, n. 3, p. 969–993, 2019. Citado 3 vezes nas páginas 32, 33 e 35.

MERRIAM-WEBSTER. *Merriam-Webster Online Dictionary*. 2018. Disponível em: <http://www.merriam-webster.com>. Citado na página 19.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. USA: Curran Associates Inc., 2013. (NIPS'13), p. 3111–3119. Disponível em: <http://dl.acm.org/citation.cfm?id=2999792.2999959>. Citado 3 vezes nas páginas 72, 73 e 86.

MINER, G. et al. *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*. 1st. ed. Orlando, FL, USA: Academic Press, Inc., 2012. ISBN 012386979X, 9780123869791. Citado na página 21.

Mittal, A. *Understanding RNN and LSTM*. 2019. Disponível em: <https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e>. Citado 2 vezes nas páginas 67 e 68.

MIYAZAKI, C. K. *Redes Neurais Convolucionais par aprendizagem e reconhecimento de objetos 3D*. 2017. Monografia (Engenharia Elétrica), São Carlos, Brazil. Citado 3 vezes nas páginas 45, 46 e 47.

MOHAMMADI, M. et al. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys Tutorials*, Institute of Electrical and Electronics Engineers (IEEE), v. 20, n. 4, p. 2923–2960, 2018. ISSN 2373-745X. Citado na página 21.

MONDAL, M.; SILVA, L. A. A.; BENEVENUTO, F. A measurement study of hate speech in social media. In: *Proceedings of the 28th ACM Conference on Hypertext and Social Media*. [S.l.]: ACM, 2017. (HT '17). Citado na página 95.

NER, S. *Stanford Named Entity Recognizer (NER)*. 2018. Disponível em: <https://nlp.stanford.edu/software/CRF-NER.html>. Citado na página 25.

NICOLAS, P. *Scala for Machine Learning*. Packt Publishing, 2015. (Community experience distilled). ISBN 9781783558759. Disponível em: <https://books.google.com.br/books?id=d5EIBgAAQBAJ>. Citado na página 107.

Olah, C. *Understanding RNN and LSTM*. 2015. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Citado na página 68.

PATCHA, A.; PARK, J.-M. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, v. 51, n. 12, p. 3448–3470, 2007. Disponível em: <http://dblp.uni-trier.de/db/journals/cn/cn51.html#PatchaP07>. Citado na página 37.

PATEL, A. *Ensemble Learning*. 2019. Disponível em: <https://medium.com/ml-research-lab/ensemble-learning-relation-with-bias-and-variance-431cdc0a3fc9>. Citado 3 vezes nas páginas 40, 41 e 43.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: *In EMNLP*. [S.l.: s.n.], 2014. Citado na página 73.

PETERS, M. E. et al. Deep contextualized word representations. In: *Proc. of NAACL*. [S.l.: s.n.], 2018. Citado na página 73.

PETROVIC, S. et al. Can twitter replace newswire for breaking news? In: *ICWSM*. [S.l.: s.n.], 2013. Citado na página 79.

PHAM, D.-S. et al. Detection of cross-channel anomalies. *Knowledge and Information Systems*, Springer-Verlag New York, Inc., v. 35, n. 1, p. 33–59, Apr 2013. Disponível em: <https://doi.org/10.1007/s10115-012-0509-6>. Citado na página 19.

PHUVIPADAWAT, S.; MURATA, T. Breaking news detection and tracking in twitter. In: *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*. Washington, DC, USA: IEEE Computer Society, 2010. (WI-IAT '10), p. 120–123. ISBN 978-0-7695-4191-4. Disponível em: <https://doi.org/10.1109/WI-IAT.2010.205>. Citado na página 25.

PIMENTEL, M. A. F. et al. Review: A review of novelty detection. *Signal Process.*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 99, p. 215–249, jun. 2014. ISSN 0165-1684. Disponível em: <http://dx.doi.org/10.1016/j.sigpro.2013.12.026>. Citado 4 vezes nas páginas 19, 20, 37 e 55.

QIAN, X. et al. Social media based event summarization by user-text-image co-clustering. *Knowledge-Based Systems*, v. 164, 11 2018. Citado 3 vezes nas páginas 31, 35 e 116.

QIAN, X. et al. Landmark summarization with diverse viewpoints. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 25, n. 11, p. 1857–1869, Nov 2015. ISSN 1051-8215. Citado na página 31.

QUINN, J. A.; WILLIAMS, C. K. I. Known unknowns: Novelty detection in condition monitoring. In: MARTÍ, J. et al. (Ed.). *Pattern Recognition and Image Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 1–6. ISBN 978-3-540-72847-4. Citado na página 37.

RANJAN, C. et al. *Dataset: Rare Event Classification in Multivariate Time Series*. 2018. Citado 2 vezes nas páginas 69 e 100.

Rey, J. *Convolutional Neural Networks for Visual Recognition*. 2019. Disponível em: <https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/>. Citado na página 48.

ROSA, H. et al. A "deeper" look at detecting cyberbullying in social networks. In: *IJCNN, 2018 International Joint Conference on Neural Networks*. [S.l.]: IEEE Explorer, 2018. p. 323–330. Citado 4 vezes nas páginas 31, 35, 84 e 106.

ROSEBROCK, A. Intersection over union (iou) for object detection. In: . [s.n.], 2016. Disponível em: <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>. Citado na página 51.

SAHIN, S.; KOZAT, S. Nonuniformly sampled data processing using lstm networks. *IEEE Transactions on Neural Networks and Learning Systems*, PP, p. 1–10, 10 2018. Citado 2 vezes nas páginas 110 e 114.

Sarang Narkhede. *Understanding AUC - ROC Curve*. 2018. Disponível em: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>. Citado na página 94.

SARWAR, A. et al. A novel method for content-based image retrieval to improve the effectiveness of the bag-of-words model using a support vector machine. *Journal of Information Science*, SAGE Publications Sage UK: London, England, v. 45, n. 1, p. 117–135, 2019. Citado na página 116.

SCHANDA, J.; SANDERSON, M.; CLOUGH, P. Examining new event detection. In: *Proceedings of the 2014 Australasian Document Computing Symposium*. New York, NY, USA: ACM, 2014. (ADCS '14), p. 26:26–26:33. ISBN 978-1-4503-3000-8. Disponível em: <http://doi.acm.org/10.1145/2682862.2682869>. Citado na página 37.

SCHINAS, M. et al. Event detection and retrieval on social media. *CoRR*, abs/1807.03675, 2018. Disponível em: <http://arxiv.org/abs/1807.03675>. Citado na página 21.

SCHUBERT, E. et al. On evaluation of outlier rankings and outlier scores. In: SIAM. *Proceedings of the 2012 SIAM International Conference on Data Mining*. [S.l.], 2012. p. 1047–1058. Citado na página 81.

Scott Fortmann-Roe. *Understanding the Bias-Variance Tradeoff*. 2018. Disponível em: <http://scott.fortmann-roe.com/docs/BiasVariance.html>. Citado 2 vezes nas páginas 41 e 42.

SEVERYN, A.; MOSCHITTI, A. Learning to rank short text pairs with convolutional deep neural networks. In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2015. (SIGIR '15), p. 373–382. ISBN 978-1-4503-3621-5. Disponível em: <http://doi.acm.org/10.1145/2766462.2767738>. Citado na página 84.

SHI, L.-l. et al. Event detection and user interest discovering in social media data streams. *IEEE Access*, PP, p. 1–1, 03 2017. Citado na página 21.

SMITH, K. *58 Incredible and Interesting Twitter Stats and Statistics*. 2019. <https://www.brandwatch.com/blog/twitter-stats-and-statistics/>. [Online; accessed 15-July-2019]. Citado na página 21.

SOCHER, R. et al. Recursive deep models for semantic compositionality over a sentiment treebank. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, 2013. p. 1631–1642. Disponível em: <https://www.aclweb.org/anthology/D13-1170>. Citado na página 74.

SOFMAN, B. et al. Anytime online novelty and change detection for mobile robots. *Journal of Field Robotics*, Wiley Online Library, v. 28, n. 4, p. 589–618, 2011. Citado na página 37.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.*, v. 45, n. 4, p. 427–437, jul. 2009. ISSN 0306-4573. Disponível em: <http://dx.doi.org/10.1016/j.ipm.2009.03.002>. Citado na página 94.

SPRINGENBERG, J. et al. Striving for simplicity: The all convolutional net. In: *ICLR (workshop track)*. [s.n.], 2015. Disponível em: <http://lmb.informatik.uni-freiburg.de/Publications/2015/DB15a>. Citado na página 46.

SUN, S. et al. Feature fusion through multitask cnn for large-scale remote sensing image segmentation. In: IEEE. *2018 10th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)*. [S.l.], 2018. p. 1–4. Citado na página 44.

TARASSENKO, L. et al. Novelty detection for the identification of masses in mammograms. In: *1995 Fourth International Conference on Artificial Neural Networks*. [S.l.: s.n.], 1995. p. 442–447. ISSN 0537-9989. Citado na página 37.

TheGuardian. *Twitter first with news of Osama bin Laden's death via ex-Bush staffer*. 2011. Disponível em: <https://www.theguardian.com/technology/blog/2011/may/02/twitter-osama-bin-laden-death-leaked>. Citado na página 79.

TUMER, K.; GHOSH, J. Error correlation and error reduction in ensemble classifiers. *Connection science*, Taylor & Francis, v. 8, n. 3-4, p. 385–404, 1996. Citado na página 59.

VADIVEL, A.; MAJUMDAR, A.; SURAL, S. Performance comparison of distance metrics in content-based image retrieval applications. In: *International Conference on Information Technology (CIT), Bhubaneswar, India*. [S.l.: s.n.], 2003. p. 159–164. Citado na página 81.

VIJAY, B.; ALEX, K.; ROBERTO, C. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 39, p. 2481–2495, 2017. Citado 2 vezes nas páginas 84 e 88.

VINYALS, O. et al. *Show and Tell: A Neural Image Caption Generator*. 2014. Cite arxiv:1411.4555. Disponível em: <http://arxiv.org/abs/1411.4555>. Citado na página 30.

WAGNER, T. et al. Semi-supervised learning on data streams via temporal label propagation. In: DY, J.; KRAUSE, A. (Ed.). *Proceedings of the 35th International Conference on Machine Learning*. Stockholmsmässan, Stockholm Sweden: PMLR, 2018. (Proceedings of Machine Learning Research, v. 80), p. 5095–5104. Citado na página 115.

WANG, S.; MINKU, L. L.; YAO, X. A systematic study of online class imbalance learning with concept drift. *IEEE Transactions on Neural Networks and Learning Systems*, v. 29, p. 4802–4821, 2018. Citado 2 vezes nas páginas 22 e 115.

YANG, Y.; PIERCE, T.; CARBONELL, J. A study of retrospective and on-line event detection. In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 1998. (SIGIR '98), p. 28–36. ISBN 1-58113-015-5. Disponível em: <http://doi.acm.org/10.1145/290941.290953>. Citado na página 37.

YONG, S.-P.; DENG, J. D.; PURVIS, M. K. Novelty detection in wildlife scenes through semantic context modelling. *Pattern Recognition*, v. 45, p. 3439–3450, 2012. Citado 4 vezes nas páginas 25, 26, 27 e 35.

YU, D.; SHEIKHOLESLAMI, G.; ZHANG, A. Findout: Finding outliers in very large datasets. *Knowl. Inf. Syst.*, Springer-Verlag New York, Inc., New York, NY, USA, v. 4, n. 4, p. 387–412, out. 2002. ISSN 0219-1377. Disponível em: <http://dx.doi.org/10.1007/s101150200013>. Citado na página 59.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. In: *Computer Vision – ECCV 2014*. [S.l.]: Springer International Publishing, 2014. p. 818–833. ISBN 978-3-319-10590-1. Citado na página 47.

ZEPHORIA, M. *The Top 20 Valuable Facebook Statistics*. 2019. Disponível em: <https://zephoria.com/top-15-valuable-facebook-statistics/>. Citado na página 109.

ZHANG, X.; LECUN, Y. *Text Understanding from Scratch*. 2015. Citado na página 72.

ZHANG, X. et al. Integrating multiple types of features for event identification in social images. *Multimedia Tools and Applications*, Springer, v. 75, n. 6, p. 3301–3322, 2016. Citado 3 vezes nas páginas 28, 29 e 35.

ZHANG, Y.; MERATNIA, N.; HAVINGA, P. J. Outlier detection techniques for wireless sensor networks: A survey. *IEEE Communications Surveys and Tutorials*, v. 12, n. 2, p. 159–170, 2010. Citado na página 37.

ZHONG, W. *Crowdsourcing novelty annotation data from tweets regarding the whaling activist event*. 2018. Disponível em: <https://github.com/CrowdTruth/Salience-In-News-And-Tweets>. Citado na página 21.

ZHOU, C. et al. A c-lstm neural network for text classification. *CoRR*, abs/1511.08630, 2015. Citado 2 vezes nas páginas 31 e 32.

ZHOU, Y. T.; CHELLAPPA, R. Computation of optical flow using a neural network. *IEEE 1988 International Conference on Neural Networks*, p. 71–78 vol.2, 1988. Citado na página 46.

ZUBIAGA, A. et al. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 51, n. 2, p. 1–36, 2018. Citado na página 33.

—