# Classificação de *Tweets* sobre Trânsito Utilizando Diferentes Técnicas de *Deep Learning*

Estevan B. Teixeira, Pedro N. de S. Moura, Carlos Alberto V. Campos
Programa de Pósgraduação em Informática (PPGI)
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Rio de Janeiro – RJ – Brazil
Email: {estevan.teixeira, pedro.moura, beto}@uniriotec.br

***Resumo*—No âmbito da mobilidade urbana, a obtenção de informação de forma rápida e localizada para tomada de decisão é um dos principais desafios atuais. Nesse contexto, redes sociais podem funcionar como uma das fontes de extração de conhecimento para diversas tarefas, dentre as quais controle de trânsito. Contudo, tais dados precisam ser bem classificados para garantir que somente informações relevantes sejam utilizadas. Particularmente em países lusófonos, não há muitos estudos sobre tal classificação, em especial explorando o potencial das redes neurais. Assim, este trabalho propõe um modelo de representação e classficação de microtexto para a língua portuguesa através de técnicas modernas de deep learning, com o objetivo de gerar informações de trânsito. Para tal, são analisados os resultados da combinação de diversas arquiteturas de deep learning para representação e classificação, levando a resultados de acurácia e precisão acima de 95%.**

## I. Introdução

Um dos maiores problemas enfrentados no dia-a-dia de grandes centros urbanos corresponde ao tempo despendido no deslocamento entre lugares. Faz-se, assim, natural investir recursos em ferramentas capazes de reduzir esse tempo de movimentação. Isso leva a uma questão particular desse problema, que corresponde à detecção de eventos de trânsito (como por exemplo: acidentes, congestionamentos, obras de manutenção nas vias ou grandes aglomerações de pessoas). Dentre diversas formas de abordar esse problema, uma das que têm se mostrado mais efetiva é coletar e partilhar informações de observadores locais.

Para tal, uma estratégia eficaz é a leitura de informações postadas em redes sociais, como o *Twitter* ou *Facebook* [1]. Contudo, tal estratégia esbarra na enorme quantidade de informações dos mais diversos tipos em tais redes [2]. Sendo assim, faz-se imperativa uma forma de diferenciar os textos entre aqueles que importam e os que não importam para o contexto de trânsito.

Estudos de classificação de texto já vêm sendo feitos há décadas [3]–[5] com o auxílio de especialistas do domínio, tais como linguistas. Contudo a ênfase recente é em modelos de aprendizado de máquina capazes de reconhecer automaticamente as características (*features*) do texto e "aprender" a classificá-lo. Particularmente, estudos recentes propõem o uso de redes neurais de aprendizado profundo, ou *Deep Learning Networks* para realizar as tarefas de classificação sem tanta necessidade de compreensão do domínio, uma vez que tais redes apresentam a capacidade de inferir as características do *dataset* analisado.

Redes sociais apresentam características bastante particulares, como textos pouco formatados, altamente coloquiais, com grande quantidade de onomatopéias, abreviações e *emojis*. Particularmente no caso dos microblogs, como o *Twitter*[1] e o *Sina Weibo*[2], características específicas incluem textos curtos e bastante diretos. Particularmente o *Twitter* é muito usado como fonte para estudos envolvendo redes sociais por sua natureza aberta, seus posts sendo públicos por definição [6], diferente de redes como o *Facebook*, que exigem permissão explícita para o acesso a posts. No entanto, estudos focados em classificação de microtextos raramente trabalham com português, sendo eles em sua maioria concentrados na língua inglesa [7]. Igualmente, há grande dificuldade em levantar bases de dados prontas de tweets em português para utilizar em tais pesquisas.

Assim, este trabalho apresenta três contribuições:

1) Uma análise de modelos de *deep learning* tanto para representar quanto para classificar *tweets* em língua portuguesa referentes à relevância no domínio de trânsito, apresentando uma comparação das principais métricas de classificação de informações (acurácia, precisão e recall) para as combinações de modelos testadas: vetorização via *Word Embedding*, em que foram usados os modelos *Word2Vec* e *GloVe*, e classificação usando redes convolucionais e recorrentes.
2) Um estudo do impacto da sensibilidade dos modelos de *deep learning* referente à variação das dimensões dos modelos de *Word Embedding* testados em um *dataset* de microtexto para a língua portuguesa.
3) Criação e disponibilização de um *dataset* de 40 mil *tweets* anonimizados em língua portuguesa, de modo a suprir a falta de bases para estudos em países lusófonos. Esse conjunto está disponibilizado livremente junto aos códigos utilizados neste artigo na plataforma *GitHub*.

O trabalho está organizado da seguinte forma. A Seção II descreve os conceitos fundamentais necessários para entender o problema investigado. A Seção III apresenta e discute os trabalhos relacionados ao tópico investigado. Por sua vez, a Seção IV apresenta a metodologia utilizada. Já a Seção V

[1]https://twitter.com/
[2]https://www.weibo.com/

aborda os experimentos realizados e os resultados obtidos. Por fim, a Seção VI descreve as conclusões e possíveis trabalhos futuros.

## II. Conceitos Preliminares

### A. Classificação

O primeiro desafio da extração de informações de um texto é reconhecer a existência de informações relevantes. Dessa forma, faz-se preciso usar alguma técnica para classificá-los. Classificação de dados é uma área importante do aprendizado de máquina, e diversas abordagens são usadas para tal fim, tais como árvores de decisão, *Support Vector Machine* e *K-Nearest Neighbors* [8].

Ao longo da última década, houve uma grande evolução nos resultados obtidos para diversos problemas de aprendizado de máquina através do uso das chamadas "redes neurais profundas", que se distinguem das redes neurais comuns por apresentarem um número considerável de camadas internas de neurônios. Tal característica possibilita uma grande melhoria no reconhecimento de padrões, em particular a detecção de detalhes sem necessidade de levantamento prévio [9]. Dentre as diversas arquiteturas de redes profundas, duas têm se mostrado particularmente interessantes para a tarefa de classificação de textos: Redes Convolucionais e Redes Recorrentes [10]–[12].

*1) Redes Convolucionais:* As Redes Convolucionais, identificadas pela sigla CNN (*Convolutional Neural Networks*), foram propostas em [13] e consistem em uma arquitetura de redes neurais composta por uma secão de extração de *features*, e uma de classificação. O uso de CNN é reconhecido como uma das principais ferramentas para detecção automatizada de features, sendo particularmente usado em processamento de imagens, mas também em processamento de linguagem natural [10].

A primeira seção da rede, seu grande diferencial, é composta por uma sequência de pares de elementos. Uma camada de convolução seguida por uma de concentração. Na camada de convolução, a matriz de entrada é submetida a um filtro, chamado de função de convolução, célula a célula. Tal filtro é aplicado sobre a célula e um conjunto de células adjacentes (formando um quadrado $3 \times 3$, $5 \times 5$, etc.). A operação corresponde a um produto interno entre o filtro e a região onde está sendo aplicado, de modo que o resultado é um único escalar, que é colocado na célula correspondente na camada de concentração. Dessa forma, as informações de um conjunto de células são concentradas em uma única, possibilitando uma melhor análise do classificador. Tal processo pode ser repetido diversas vezes, adicionando-se pares de camadas de convolução e concentração.

A segunda seção da rede é responsável pela classificação dos dados. Geralmente essa seção é composta por: (i) uma camada de achatamento, que reduz as dimensões das informações obtidas a partir da camada de convolução, (ii) uma camada de neurônios *fully-connected*, para pontuar os elementos e (iii) uma camada de classificação, em geral usando a função *softmax*, que seleciona a classe com maior semelhança aos resultados da rede [14].

*2) Redes Recorrentes:* Já as redes recorrentes, conhecidas pela sigla RNN (*Recurrent Neural Networks*), são uma arquitetura de redes profundas em que o resultado do passo $n$ é usado como entrada para o passo $n + 1$. Dessa forma, a camada de recorrência alimenta a si mesma, justificando o nome. Ao fazer isso, a rede carrega uma "memória" dos eventos passados, que influenciam nos resultados seguintes. São bastante eficazes no reconhecimento de entidades nomeadas e *speech tagging*, situações em que o contexto carrega informação necessária ao reconhecimento [11].

Uma questão importante sobre a arquitetura RNN tradicional é o fato de que as informações não são "esquecidas", fazendo com que os resultados iniciais influenciem todo o processo. Para abordar essa questão, foi proposta a rede *Long Short-Term Memory Network* (LSTM, ou rede de memória longa de curta-duração) [15]. Nessa arquitetura, a recorrência é composta por quatro camadas de ativação, ou "portões" que a cada novo passo, selecionam se:

1) A informação contida no neurônio deve ser esquecida (*Forget Gate*);
2) A informação recebida deve ser lembrada (*Input Gate*); e
3) A informação recebida deve ser enviada para a função de ativação (*Output Gate*).

A quarta camada é a função de ativação padrão da rede neural, que computa a informação e envia para o classificador. Dessa forma, dependendo das condições, uma informação ao entrar na rede pode demandar que a "memória" seja apagada, não sendo avaliada com base nas informações passadas [9].

### B. Representação

O uso de tais ferramentas para classificação de textos geralmente demanda a representação dos mesmos em vetores numéricos, uma vez que a maioria dos classificadores, particularmente redes neurais, são focados no trabalho com números. Nas próximas seções, serão brevemente descritas algumas técnicas de representação, desde as mais básicas, como Bag-of-Words e Tf-Idf, até mais avançadas, como aquelas baseadas em *Word Embedding*, tais como *Word2Vec* e *GloVe*.

*1) Bag-of-words:* As técnicas básicas de vetorização são coletivamente chamadas de *Bag-of-Words*, por essencialmente armazenarem todas as palavras de um docuemnto em uma "bolsa", desprezando sua posição relativa. O método mais simples consiste em basicamente contar as ocorrências de uma palavra no texto. Nesse sistema, cada palavra presente no *corpus* como um todo é considerada uma dimensão de um vetor, e o seu valor será o número de vezes que a palavra aparece. Assim, um documento é representado por um vetor em um espaço $n$-dimensional, em que $n$ é o número de palavras no *corpus* [16].

Uma evolução desse método é o chamado *Term Frequency/Inverse document frequency* (Tf-Idf). Nesse método, a freqüência dos termos em seus documentos é normalizada pelo inverso de sua frequência no *corpus*, segundo a fórmula $w_{x,y} = tf_{x,y} * log(\frac{N}{df_x})$. Assim como na contagem básica, esse método tende a gerar vetores esparsos e, apesar de carregar

alguma informação semântica, o mesmo não considera as semelhanças entre os termos, além de obviamente ignorar suas posições.

*2) Word Embeddings:* Para atacar tais problemas, são usados métodos de *deep learning* para vetorização dos termos, os quais têm por objetivo projetar as palavras em um espaço n-dimensional, através do uso de redes profundas, onde proximidade vetorial se traduza em proximidade semântica. Esses sistemas são coletivamente chamados de *Word Embeddings*.

*Word2Vec* foi uma das primeiras propostas de *Word Embeddings*, desenvolvidas por um grupo de cientistas da Google [17], e compreende duas implementações distintas do conceito. A primeira, *Continuous Bag-Of-Words*, ou CBOW, recebe uma sequência de palavras, buscando prever uma palavra (geralmente de posição central) do grupo, a partir das outras e, no processo, gera sua a representação vetorial. A outra implementação, *Skip-Gram*, propõe o oposto, tentando obter o contexto a partir da palavra analisada. Em ambos os casos, o conjunto de palavras é submetido a um *AutoEncoder*, e o resultado da rede é comparado com o gabarito, utilizando uma função de *back-propagation* para aprender os padrões.

*Global Vectors*, ou *GloVe*, são uma terceira implementação de *Word Embeddings*, proposta por uma equipe da universidade de Stanford [18]. Essa implementação identifica as palavras por meio de uma matriz de co-ocorrência, que avalia a probabilidade de duas palavras aparecerem no mesmo texto. Seus resultados são similares ao Word2Vec, embora com uma distribuição vetorial mais abrangente.

## III. Trabalhos relacionados

Em [19] é apresentada uma metodologia para combinar dados de pontos de GPS, tweets e informações públicas sobre vias urbanas e clima para construir um mapa em tempo quase-real dos congestionamentos em Chicago. Um conjunto de pontos de GPS é mapeado de modo a revelar regiões espaço-temporais de congestionamento. Esse mapa é combinado com dados do Twitter, obtidos de duas formas: 1) Tweets de autoridades públicas e usuários focados em informações de trânsito, formatados segundo regras bastante específicas e recuperáveis através do uso de expressões regulares; e 2) Tweets do público geral, identificados por palavras-chave e cujas informações espaciais também são recuperadas via expressões regulares. Tais tweets são geocodificados (nome dado ao processo de transformar endereços em coordenadas de mapa) e aplicados ao mapa. O resultado é um conjunto de congestionamentos com tweets associados por similaridade espaço-temporal que trazem informações adicionais, não sendo capaz de detectar outros tipos de evento, nem usar a rede social como fonte primária.

Baseado nessa iniciativa, [20] propõe um framework para extrair dados de mídias sociais (especificamente Weibo-Sina), relacionados a eventos de trânsito esporádicos. O trabalho propõe o uso de dados de GPS de taxi locais para definir situações comuns e removê-las do escopo. O trabalho detecta eventos no twitter geocodificando entidades nomeadas e pareando-as espaço-temporalmente com as anomalias de deslocamento dos taxis. Diferente do trabalho anterior, isso significa que é possível detectar diversos tipos de eventos, embora os tweets ainda não sejam a fonte primária. Além disso, um ponto a ser mencionado é que o uso de entidades nomeadas é limitado a localizações a geocodificar, significando que o sistema não depende de palavras-chave para identificação de eventos.

Igualmente, em [21] foi construido uma aplicação que une dados veiculares com mídias sociais para mapear eventos relevantes relacionados ao trafego, como shows, acidentes, locais para hospedagem, etc. Sua abordagem foi oposta aos anteriores, partindo de incidentes relatados em sistemas de mapeamento colaborativo e buscando localizar tweets em uma dimensão espaço-temporal similar. Assim, o trabalho é incapaz de detectar um evento a partir de um tweet, necessitando que o sistema já o conheça previamente. Além disso, o uso de palavras-chave para classificar a relevância de uma postagem apresenta uma dificuldade com relação a erros de digitação e palavras desconhecidas.

No mesmo ano, em [22] foi proposto o uso de um grafo de visibilidade, modelando tweets pré-classificados como uma distribuição de Poisson, sendo assim capazes de detectar eventos conhecidos e desconhecidos relacionados ao contexto sem necessidade de pré-definir palavras chave. Tais eventos são agrupados por meio de grafos de visibilidade, onde cada vértice é uma palavra e as arestas representam a conexão de duas palavras em um bigrama, e sua partição é realizada através de um processo de clusterização Markoviano. Devido à necessidade de pré-classificação dos tweets, o sistema meramente particiona os eventos em um conjunto relevante, sendo portanto inviável para uso stand-alone em tempo real. Contudo, sua combinação com um classificador ágil poderia resolver tal pendência.

Além disso, foi apresentado em [7] um *framework* para a detecção de eventos relacionados a trânsito na cidade do Rio de Janeiro a partir de *tweets* em língua portuguesa, utilizando como classificador o modelo *Conditional Random Fields*. As entidades são reconhecidas e classificadas a partir do conjunto de ferramentas *Stanford Named Entity Recognition* (Stanford NER). Foram extraídos *tweets* tanto do público geral quanto de perfis associados a órgãos governamentais. Contudo, a dificuldade na reprodução dos passos devido ao uso de códigos não documentados no artigo o torna de difícil comparação com outros métodos.

Finalmente, em [12] desenvolveram um sistema em que classifica tweets em relevantes ou não para trânsito, utilizando uma combinação de redes convolucionais e recorrentes, e apresentando elevado grau de precisão. Tal metodologia faz uso de ferramentas comuns de machine learning, como o sistema *TensorFlow*, da Google, para construir suas redes. Apesar do elevado grau de sucesso, o trabalho aparenta estar em fase inicial, pois não executa certas comparações e testes, como o uso de vetorização GloVe como comparativo ao Word2Vec e fastText aplicados, além de não ser testado em outras linguagens além do inglês.


Figura 1. Sequência de etapas do modelo proposto para a representação e classificação de microtextos.

## IV. Modelo proposto

Nesta seção é apresentado o modelo proposto para o problema da representação e classificação de microtextos em língua portuguesa no contexto do trânsito urbano. Para isso, o modelo proposto faz uso de técnicas de aprendizado de máquina para a vetorização e classificação por meio de redes profundas e, é importante dizer que, este modelo é baseado no trabalho descrito em [12].

O modelo é composto pela seguintes etapas: entrada, pré-processamento, vetorização, classificação e saída dos dados na forma de textos relevantes. A Figura 1 ilustra estas etapas do modelo proposto, sendo que as mesmas serão descritas em detalhes a seguir.

### A. Entrada e pré-processamento

O sistema recebe como entrada uma lista de tweets gravados em arquivo CSV (Comma Separated Values, ou valores separados por vírgula), e os pré-processa removendo caracteres especiais, transformando o texto em caixa baixa e convertendo-o em uma lista de palavras, ou *tokens*. No processo também é removido um conjunto de *stopwords*, de modo a aumentar o valor semântico dos *tokens*.

### B. Vetorização

Após o pré-processamento, os conjuntos de *tokens* são transformados em matrizes $n \times m$, onde $n$ é o número de *tokens* no texto e $m$ o número de dimensões pré-definidas no vetorizador. O processo utiliza a chamada pré-vetorização, buscando as palavras em um grande dicionário previamente montado e substituindo-as pelo vetor equivalente, ou por um vetor de zeros, caso não seja encontrada.

### C. Classificação

A classificação foi feita segundo o algoritmo de [12], utilizando três implentações: CNN pura, LSTM pura e uma combinação de ambas. A combinação de ambas as redes se justifica pela complementação da extração de *features* aprimorada das redes convolucionais e a aplicação em dados sequenciais, como sentenças sendo sequencias de palavras, atacada pelas redes recorrentes. Tal abordagem vem encontrando aplicações cada vez mais comuns na área de NLP [10], [11].

Nesse ponto, optou-se por seguir os hiper-parâmetros defenidos no trabalho original, como o número de filtros e unidades de memória, como explicado na Seção V, uma vez que já estão estabelecidos como melhor funcionamento. Apesar disso, foram incorporadas novas variáveis no processo, de modo a estudar o seu impacto no resultado final.

No caso da combinação das redes, os dados são processados sequencialmente por uma CNN e em seguida por uma LSTM, sem o uso de camada de acumulação entre elas, como mostrado na Figura 2, que mostra os conjuntos de *tokens* em suas janelas, sendo submetidos à função de convolução e alimentando, cada janela, um passo da recorrência.

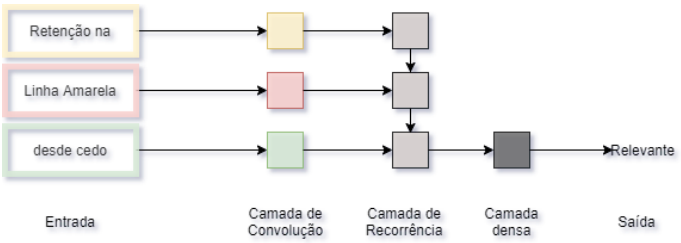



Figura 2. Modelo CNN + LSTM

### D. Saída

Os resultados são submetidos a um classificador *softmax*, e a saída final é a classificação binária do texto em relevante ou não a eventos de trânsito. Assim, os textos classificados online como relevantes poderão ser utilizados em um sistema online de informações de trânsito.

## V. Experimentos e Resultados

Com o objetivo de evidenciar o uso do modelo proposto, nesta seção é realizado um estudo de como extrair conhecimento de microtextos a respeito das condições do trânsito de centros urbanos. Para isso, foram realizados experimentos sobre a rede social Twitter em execução no Brasil. Nas próximas subseções, serão descritas detalhadamente as tarefas a respeito da coleta e etiquetamento do *dataset* usado nos experimentos, do processamento realizado e sobre os resultados obtidos.

### A. Coleta e etiquetamento do Dataset

Para os experimentos, foi coletado um *dataset* composto por 43.152 *tweets*, os quais todos são do idioma português brasileiro (*tag languages* contendo o valor pt-BR), e filtrados pela área geográfica de um retângulo localizado entre as latitudes $5,88$ N e $34,31$ S e longitudes $34,97$ W e $74,62$ W, correspondente à região do Brasil. Além destes filtros na API de coleta, o sistema também foi programado para descartar *tweets* que fossem compostos por menos de quatro palavras, após o descarte de sinais de pontuação, quebras de linha, urls e *stopwords*. Dentre os *tweets* obtidos, 18.74 foram coletados do histórico postado por perfis tradicionalmente vinculados ao domínio de trânsito até 13 de março de 2020, sendo estes: @OperacoesRio, @CETRIO_ONLINE, @transitorj, @CETSP_, @TransitoSampaSP e @radiotransitofm. Já os outros 24.359 foram obtidos pelo método *streaming*, sendo tweets postados em 14 de março de 2020 que atendiam aos filtros supracitados.

Os textos obtidos foram classificados manualmente, após uma pré-classificação por meio de um dicionário das palavras mais frequentes nos *tweets* dos perfis @CETRIO_ONLINE e @CETSP_. Dessa forma, chegou-se a um conjunto de 16.315

*tweets* relevantes ao domínio de trânsito, e outros 26.837 não-relevantes. O *dataset* foi então dividido aleatoriamente em conjuntos de treino e teste, através do uso da ferramenta *scikit-learn*, respeitando uma proporção de 4:1 para treino e teste, respectivamente.

### B. Processamento

O pré-processamento fez uso de funções nativas da linguagem *Python*, bem como da lista de *stopwords* em português provida pelo pacote NLTK [23]. Os *tweets* foram vetorizados a partir do uso de dicionários pré-vetorizados, os quais foram gerados pelas metodologias detalhadas em [24]. Foram usados os dicionários de Word2Vec/SkipGram e GloVe, que foram adapatados para a língua portuguesa e disponibilizados pelos autores[3]. Os dicionários apresentam vetorizações em 50, 100, 300, 600 e 1000 dimensões, variável essa usada na pesquisa para determinar o conjunto mais adequado de dimensões. Tais vetores são truncados no 30º token, levando a um *input* para o classificador de uma matriz 30 vezes o número de dimensões do vetorizador.

Tal truncamento foi feito de modo a garantir um *input* de tamanho fixo, necessário para as redes implementadas no pacote *Keras*. A escolha do número 30 se deu mediante uma análise do número de palavras dos *tweets*, sendo que tal número contempla 98.5% dos textos, após a remoção das *stopwords* e *urls*, como mostrado na Figura 3, sem causar *overflows* de memória.

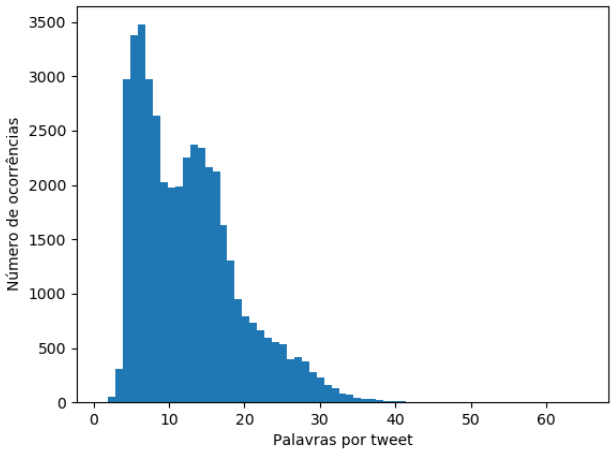
A classificação foi feita através dos três modelos propostos por [12]. O primeiro modelo utiliza uma rede CNN com 200 filtros e janela de convolução composta por 2 vezes o número de dimensões do vetorizador, aplicando a funcão convolucional a cada duas palavras, e sobre todas as dimensões de cada palavra. A rede utiliza ativação por *relu* e uma camada de agrupamento baseada no output padrão da CNN. Tal configuração utiliza um *dropout* de 0.5, descartando aleatoriamente metade dos *inputs*.

O segundo modelo utilizou uma rede LSTM, com 50 unidades e funções padrão de ativação (*tanh* para a ativação comum e *sigmoid* para a recorrência). Da mesma forma que no modelo anterior, usa-se um *dropout* de 0.5. Finalmente, o terceiro modelo utiliza uma combinação de ambas as redes, submetendo o *input* primeiro a um CNN nos mesmos moldes do primeiro modelo, exceto pelo uso de um *dropout* de 0.25, e sem camada de agrupamento. O resultado foi, então, submetido a uma rede LSTM de 100 unidades, com suas outras configurações idênticas ao segundo modelo, inclusive o *dropout*.

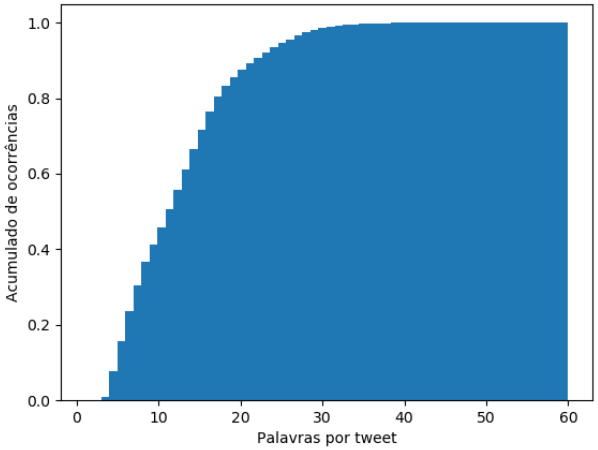
Em todos os casos foram usados o otimizador ADAM e a classificação por *softmax*. O código utilizado encontra-se disponível no repositório *GitHub*[4].

[3]http://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc

[4]https://github.com/EBarbara/twitter-research




(a) Histograma da quantidade de palavras




(b) Distruição cumulativa da quantidade de palavras

Figura 3. Análise da quantidade de palavras por *tweet* do dataset analisado.

### C. Resultados

Os resultados dos experimentos foram compilados nas Tabelas I e II, que mostram os resultados alcançados nos experimentos para o vetorizador Word2Vec e GloVe, respectivamente, e ilustrados de forma gráfica nas Figuras 4 e 5, que mostram visualmente os resultados em cada uma das métricas (acurácia, precisão e *recall*) versus o número de dimensões dos modelos Word2Vec e GloVe, respectivamente.

*1) Análise de Sensibilidade:* Foi realizada uma análise de sensibilidade para o comportamentos dos modelos a partir da variação do número de dimensões dos vetorizadores. Primeiramente, estabelece-se que o tempo de vetorização corresponde ao tempo entre o início da carga dos modelos pré-treinados até a saída da função que substitui os *tokens* pelos valores vetoriais, enquanto o tempo de treinamento é definido como o tempo gasto na função de treinamento da rede classificadora. Dessa forma, tanto o tempo de vetorização quanto o de

Tabela I
RESULTADOS DOS EXPERIMENTOS COM VETORIZAÇÃO *Word2Vec*.

| dimensões | classificador | tempo vetorização | tempo treinamento | acurácia | precisão | *recall* | medida f1 |
|---|---|---|---|---|---|---|---|
| 50 | CNN | 53.46 | 62.50 | 96.56% | 95.77% | 95.00% | 95.38% |
| 100 | CNN | 97.77 | 75.12 | 96.80% | 96.24% | 95.16% | 95.70% |
| 300 | CNN | 275.26 | 141.89 | 96.84% | 96.21% | 95.29% | 95.75% |
| 600 | CNN | 539.29 | 247.30 | 96.87% | 95.89% | **95.72%** | 95.81% |
| 1000 | CNN | 898.13 | 376.35 | 96.86% | 95.96% | 95.62% | 95.79% |
| 50 | LSTM | 52.78 | 174.81 | 96.51% | 96.10% | 94.50% | 95.29% |
| 100 | LSTM | 98.67 | 198.74 | 96.76% | 96.64% | 94.62% | 95.62% |
| 300 | LSTM | 276.74 | 571.93 | 96.69% | 96.30% | 94.80% | 95.54% |
| 600 | LSTM | 539.99 | 1584.08 | 96.72% | 96.55% | 94.61% | 95.57% |
| 1000 | LSTM | 920.43 | 3072.39 | 96.78% | 96.23% | 95.11% | 95.66% |
| 50 | CNN+LSTM | 54.18 | 1774.15 | 96.84% | 96.40% | 95.09% | 95.74% |
| 100 | CNN+LSTM | 96.82 | 384.49 | 97.00% | **96.79%** | 95.13% | **95.95%** |
| 300 | CNN+LSTM | 270.28 | 475.51 | 96.91% | 96.34% | 95.37% | 95.85% |
| 600 | CNN+LSTM | 535.34 | 641.35 | 96.89% | 95.98% | 95.68% | 95.83% |
| 1000 | CNN+LSTM | 866.16 | 825.71 | **96.91%** | 96.42% | 95.26% | 95.83% |

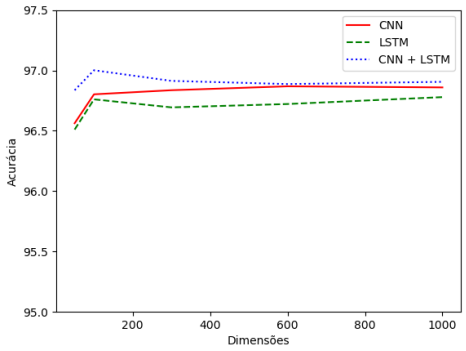
Tabela II
RESULTADOS DOS EXPERIMENTOS COM VETORIZAÇÃO *GloVe*.

| dimensões | classificador | tempo vetorização | tempo treinamento | acurácia | precisão | *recall* | medida f1 |
|---|---|---|---|---|---|---|---|
| 50 | CNN | 50.98 | 73.98 | 96.59% | 95.69% | 95.17% | 95.42% |
| 100 | CNN | 95.50 | 92.35 | 96.74% | 96.27% | 94.94% | 95.60% |
| 300 | CNN | 269.23 | 160.87 | 96.86% | 96.09% | **95.48%** | 95.78% |
| 600 | CNN | 530.30 | 270.13 | 96.77% | 96.33% | 94.98% | 95.65% |
| 1000 | CNN | 888.83 | 401.36 | 96.72% | 96.46% | 94.68% | 95.56% |
| 50 | LSTM | 54.03 | 195.68 | 96.60% | 95.72% | 95.17% | 95.44% |
| 100 | LSTM | 97.99 | 239.34 | 96.69% | 96.26% | 94.83% | 95.54% |
| 300 | LSTM | 275.62 | 597.26 | 96.79% | 96.29% | 95.07% | 95.67% |
| 600 | LSTM | 542.25 | 559.82 | 96.73% | 96.56% | 94.62% | 95.58% |
| 1000 | LSTM | 887.57 | 1012.41 | 96.74% | 96.37% | 94.84% | 95.60% |
| 50 | CNN+LSTM | 51.56 | 408.55 | 96.78% | **96.61%** | 94.70% | 95.64% |
| 100 | CNN+LSTM | 94.64 | 510.96 | 96.80% | 96.46% | 94.91% | 95.68% |
| 300 | CNN+LSTM | 269.79 | 730.64 | 96.85% | 96.36% | 95.16% | 95.75% |
| 600 | CNN+LSTM | 525.97 | 952.45 | **96.93%** | 96.47% | 95.27% | **95.86%** |
| 1000 | CNN+LSTM | 862.48 | 1197.73 | 96.89% | 96.21% | 95.43% | 95.82% |

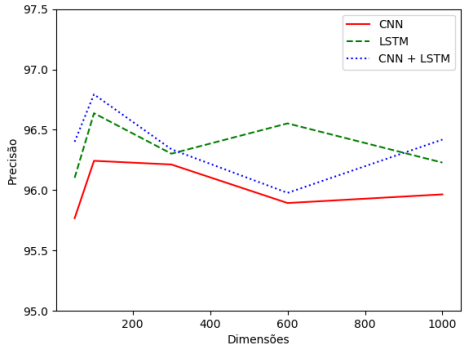
treinamento crescem em função do número de dimensões do *word embedding*. Particularmente, o tempo de vetorização permanece praticamente linear com o número de dimensões. Já o tempo de treinamento das redes apresenta uma derivada decrescente, embora ainda cresça baseado no número de dimensões. É perceptível a interferência de variáveis referentes ao uso do processador e da memória no tempo de treinamento da rede. Particularmente, quanto maior a sequência de acessos ao *TensorFlow* sem limpeza da memória, mais lentos ficam os treinamentos seguintes, como evidenciado na diferença dos tempos para a rede CNN + LSTM para os vetorizadores Word2Vec e GloVe. Além disso, a observação dos resultados não demonstra tendência de aumento ou redução nas medidas de acurácia, precisão e *recall* em função do número de dimensões, como pode ser observado nos gráficos mostrados nas Figuras 4 e 5. A escala usada, ajustada, entre 95% e 97%, aponta para um pequeno impacto da variação de dimensões.

*2) Classificadores:* De modo geral, a combinação CNN + LSTM possui performance superior às outras, em todas as métricas e combinações de vetorizador e dimensão, como esperado devido a combinar as melhores características de redes convolucionais e recorrentes. Tal modelo apresenta também os maiores tempos de treinamento, seguindo as expectativas por se tratar de uma combinação particularmente complexa de redes. Apesar disso, em diversas combinações, ocorrem situações de superioridade de outras redes. Novamente, a pequena escala da variação dos resultados aponta para que na prática as redes sejam equivalentes e a aleatoriedade dos resultados das redes seja mais significativo que a própria diferença entre as redes.

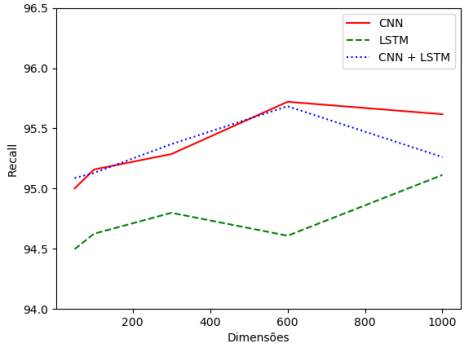
*3) Vetorizadores:* Os dois vetorizadores apresentam performances similares em todas as análises de acurácia, precisão e recall, sendo o Word2Vec marginalmente superior. A Figura 6 apresenta um comparativo das métricas versus o número de dimensões, tendo como análise o desempenho de cada vetorizador para o classificador CNN + LSTM, que havia sido considerado com melhores resultados. A figura demonstra a performance bastante próxima dos modelos, sem revelar uma melhoria particularmente perceptível entre um e outro.




(a) Acurácia




(b) Precisão

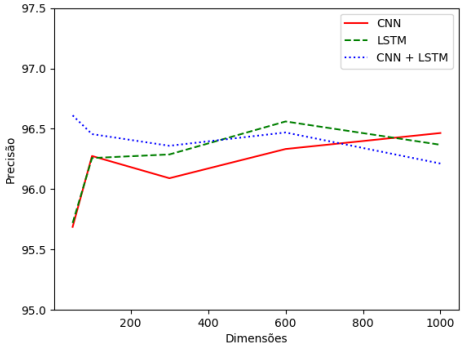



(c) *Recall*

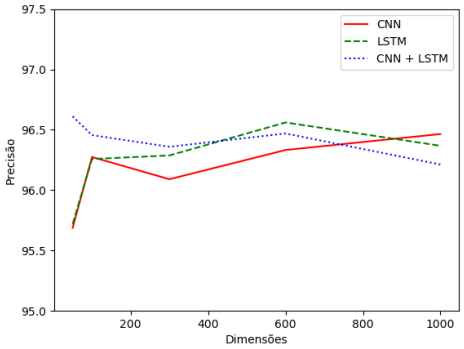
Figura 4. Comparação dos modelos CNN, LSTM e CNN+LSTM utilizando vetorização Word2Vec, exibindo dimensão versus outra métrica: a) Acurácia; b) Precisão; e c) *Recall*.

## VI. CONCLUSÃO

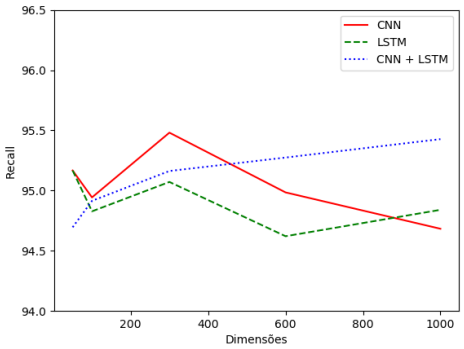
Neste trabalho, foi estudado o uso de técnicas de *deep learning* para a representação e classificação de microtextos,




(a) Acurácia

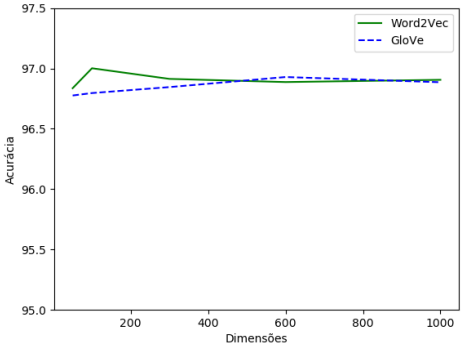



(b) Precisão

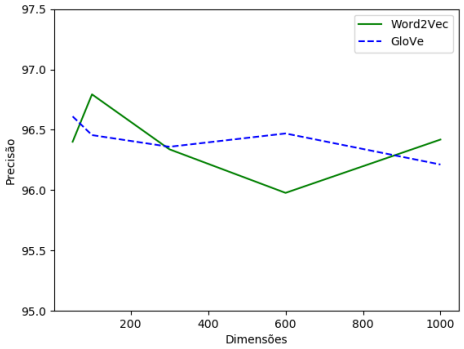



(c) *Recall*

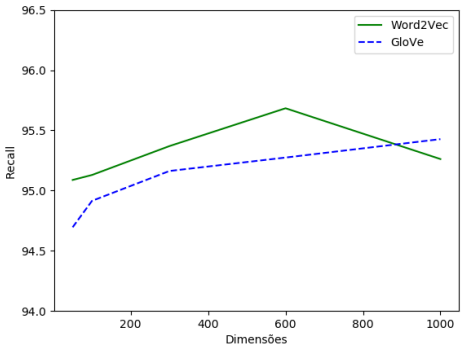
Figura 5. Comparação dos modelos CNN, LSTM e CNN+LSTM utilizando vetorização GloVe, exibindo dimensão versus outra métrica: a) Acurácia; b) Precisão; e c) *Recall*.




(a) Acurácia




(b) Precisão




(c) Recall

Figura 6. Comparação dos modelos Word2Vec e GloVe utilizando classificação CNN+LSTM, exibindo dimensão versus outra métrica: a) Acurácia; b) Precisão; e c) *Recall*.

considerando a relevância ao domínio de trânsito. Os experimentos buscaramm avaliar modelos bem conhecidos de *word embeddings* e classificadores aplicados à lingua portuguesa, bem como avaliar o impacto da variação dimensional em diferentes combinações desses modelos. Através deles pudemos avaliar a equivalência dos vetorizadores Word2Vec e GloVe, bem como a superioridade da combinação de redes CNN e LSTM para a classificação dos textos, apresentando um *trade-off* entre o tempo gasto na carga e treinamento da rede, e a melhoria das métricas de acurácia, precisão e *recall*. Da mesma forma é perceptível o pouco ganho no aumento de dimensões dos vetorizadores, em comparação com o tempo de carga do *input*, sendo seguro afirmar que 50 ou 100 dimensões bastam para uma análise eficiente.

Planeja-se em trabalhos futuros a incorporação de outros vetorizadores, como *fastText* e *Google BERT*, e classificadores, como a rede GRU, bem como o uso de métodos de redução da dimensionalidade de modo a reduzir o impacto dos grandes tempos gastos na vetorização. Além disso, pretende-se incorporar o método a uma ferramenta de georreferenciamento, de

modo a construir uma solução completa que possa ser usada por motoristas para melhorar suas experiências no trânsito.

## REFERÊNCIAS


[1] S. K. Endarnoto, S. Pradipta, A. S. Nugroho, and J. Purnama, "Traffic condition information extraction & visualization from social media twitter for android mobile application," in *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*. IEEE, 2011, pp. 1–4.

[2] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas, "Short text classification in twitter to improve information filtering," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 2010, pp. 841–842.

[3] E. Riloff and W. Lehnert, "Information extraction as a basis for high-precision text classification," *ACM Transactions on Information Systems (TOIS)*, vol. 12, no. 3, pp. 296–333, 1994.

[4] H. Ragas and C. H. Koster, "Four text classification algorithms compared on a dutch corpus," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998, pp. 369–370.

[5] J. Fürnkranz, "Exploiting structural information for text classification on the www," in *International Symposium on Intelligent Data Analysis*. Springer, 1999, pp. 487–497.

[6] "Termos de serviço do twitter," acessado: 16 de agosto de 2020. [Online]. Available: https://twitter.com/pt/tos

[7] L. Teteo, P. Moura, E. Soares, and C. Campos, "Um framework de extração e etiquetamento de informações de trânsito," in *Anais do XVIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação*. Porto Alegre, RS, Brasil: SBC, 2019. [Online]. Available: https://portaldeconteudo.sbc.org.br/index.php/wperformance/article/view/6472

[8] B. Y. Pratama and R. Sarno, "Personality classification based on twitter text using naive bayes, knn and svm," in *2015 International Conference on Data and Software Engineering (ICoDSE)*. IEEE, 2015, pp. 170–174.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[10] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[11] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 357–370, 2016.

[12] S. Dabiri and K. Heaslip, "Developing a twitter-based traffic event detection model using deep learning architectures," *Expert Systems with Applications*, vol. 118, pp. 425–439, 2019.

[13] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[14] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[16] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson Education, 2006.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[18] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[19] S. Wang, X. Zhang, J. Cao, L. He, L. Stenneth, P. S. Yu, Z. Li, and Z. Huang, "Computing urban traffic congestions by incorporating sparse gps probe data and social media data," *ACM Transactions on Information Systems (TOIS)*, vol. 35, no. 4, pp. 1–30, 2017.

[20] Z. Zheng, C. Wang, P. Wang, Y. Xiong, F. Zhang, and Y. Lv, "Framework for fusing traffic information from social and physical transportation data," *PloS one*, vol. 13, no. 8, 2018.

[21] P. H. L. Rettore, I. Araujo, J. G. M. de Menezes, L. Villas, and A. A. F. Loureiro, "Serviço de detecção e enriquecimento de eventos rodoviários baseado em fusão de dados heterogêneos para vanets," in *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC, 2019, pp. 363–376.

[22] F. F. A. Tenorio, E. Chagas, P. Barros, and H. S. Ramos, "Detecção de eventos no twitter através de grafos de visibilidade natural," in *Anais do III Workshop de Computação Urbana*. SBC, 2019, pp. 181–193.

[23] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[24] N. S. Hartmann, E. R. Fonseca, C. D. Shulby, M. V. Treviso, J. S. Rodrigues, and S. M. Aluísio, "Portuguese word embeddings: Evaluating on word analogies and natural language tasks," in *Anais do XI Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana*. SBC, 2017, pp. 122–131.