Identificando e Recuperando Anomalias em Sistemas de Veículos com Rotas

Leonardo Alves de Melo[1], **Luis F. Gomez Gonzalez**[2], **Juliana Freitag Borin**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6.137 – 13081-970 – Campinas – SP – Brazil

[2]Konker Labs
Av. Brg. Faria Lima, 1931, Sala 92 – 01451-001 – São Paulo – SP – Brazil

l156188@dac.unicamp.br, juliana@ic.unicamp.br, luis@konkerlabs.com

Abstract. *One of the expected applications for the Internet of Things is the intelligent management and maintenance of public bus lines. In this context, the monitoring systems of these vehicles need algorithms that identify in real time possible inconsistencies in routes, being able to circumvent this type of error as soon as possible, maintaining the quality of service. This paper proposes algorithms that infer in real time paths of vehicles that have a predefined route. The proposed algorithms were analyzed and compared with a baseline solution using data from a real system.*

Resumo. *Uma das aplicações esperadas para a Internet das Coisas é a gestão e manutenção inteligente de linhas de ônibus públicos. Nesse contexto, os sistemas de monitoramento desses veículos necessitam de algoritmos que identifiquem em tempo real possíveis inconsistências em trajetos, podendo o quanto antes contornar esse tipo de erro, mantendo a qualidade do serviço. Este artigo propõe algoritmos que inferem em tempo real trajetos de veículos que possuem uma rota predefinida. Os algoritmos propostos foram analisados e comparados com um algoritmo de referência utilizando dados de um sistema real.*

1. Introdução

Segundo a Pesquisa Nacional por Amostra de Domicílios (PNAD), 84,72% da população brasileira vive em áreas urbanas [IBGE], isso significa que propostas que melhorem a mobilidade nessas regiões têm um grande impacto para a qualidade de vida da maior parte da população. A presença de várias *coisas* capazes de se comunicar com a Internet e entre si para atingir um objetivo em conjunto, como por exemplo sensores, atuadores e celulares, formam o conceito de *Internet das Coisas*, do inglês *Internet of Things* (IoT). O impacto disso no dia-a-dia das pessoas se dá desde melhorias no transporte e saúde até otimização de processos industriais [Atzori et al. 2010]. Quando se trata de mobilidade urbana, há muitos trabalhos que exploram a potencialidade da Internet das Coisas e da análise de dados para coletar informações importantes tanto para governos quanto para cidadãos [Chen et al. 2014, Martins and Cunha 2018, Nunes et al. 2018, Viana et al. 2019].

Na Unicamp, o projeto *Smart Campus* tem investigado o uso de IoT para melhorar os serviços na universidade. Nesse contexto, criou-se um Sistema de Monitoramento de Ônibus Inteligente, o Circulino, que permite monitorar os veículos de transporte público

no campus de Barão Geraldo por meio de dispositivos IoT instalados nos ônibus, possibilitando aos usuários desse transporte observar em tempo real por meio de um aplicativo as posições dos ônibus de cada linha [Barbosa et al. 2019]. Embora soluções de mobilidade baseadas em IoT ajudem a melhorar serviços de transporte em cidades, falhas nesses sistemas podem resultar em transtornos na rotina do usuário. Tomando como exemplo o Circulino, existe uma falha reportada pela Prefeitura Universitária que faz com que um ônibus de uma linha apareça no aplicativo como se fosse de outra, prejudicando a experiência dos usuários que usam o aplicativo. Esse erro começa pouco antes do início de um trajeto, no momento em que é configurado equivocadamente no sistema qual a rota que o motorista irá realizar, conforme descrito na Seção 3. Em contato com órgãos e empresas que oferecem esse tipo de serviço em universidades e cidades, foi possível averiguar que esse tipo de falha é comum.

Este artigo propõe uma solução para identificar e recuperar anomalias em sistemas de veículos com rotas e apresenta uma análise comparativa com um algoritmo de referência usado por uma empresa privada que oferece esse tipo de monitoramento para cidades. O restante do documento está organizado da seguinte forma: a Seção 2 discute os trabalhos relacionados; a Seção 3 apresenta a fundamentação para o entendimento do problema; a Seção 4 apresenta a metodologia utilizada neste projeto; a Seção 5 apresenta os resultados alcançados e a Seção 6, as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Visando melhorar o transporte universitário entre campi, [Sousa et al. 2019] propõem um sistema baseado em um dispositivo IoT acoplado nos ônibus. O dispositivo contém um módulo de GPS (*Global Positioning System*) e um comunicador GSM (*Global System for Mobile Communications*), coleta dados de latitude, longitude, hora e velocidade e os envia para uma plataforma com um banco de dados. Foram coletados dados durante 5 dias de uma rota entre dois campi, e, utilizando algoritmos de aprendizado de máquina, foi estimado o tempo de chegada do veículo em cada ponto do trajeto, permitindo exibir aos usuários informações mais precisas sobre a chegada do ônibus.

Para minimizar dados incorretos no transporte inteligente da cidade de São Paulo, [Neves 2019] utilizou dados obtidos por meio de *crowdsourcing*, ou seja, gerados por um grupo de pessoas para complementar ou corrigir informações existentes no transporte público. Os dados de *crowdsourcing* são gerados pelo aplicativo *Coletivo* e utilizados em um modelo de aprendizado de máquina para prever quais contribuições são confiáveis.

[Miranda et al. 2017] desenvolveram um dispositivo que registra dados de latitude e longitude do ônibus. O diferencial desse trabalho é que ele também registra um trajeto bem definido, guardando posições desde o início até o final do caminho. [Saad et al. 2018] também desenvolveram um sistema que monitora o posicionamento dos ônibus e prediz quanto tempo falta para o ônibus chegar na próxima parada.

A análise proposta por [Gong et al. 2013] permitiu a predição do tempo de chegada de ônibus em cada ponto utilizando como base dados do histórico das leituras de dispositivos GNSS (*Global Navigation Satellite System*) presentes em cada veículo. Os pesquisadores desenvolveram um modelo híbrido de predição que foi testado em Shenyang, China, e obteve resultados melhores do que a tabela predefinida de horários. [Simmons et al. 2006] conseguiu prever com 98% de certeza a rota que um determinado

motorista irá realizar tomando como base suas viagens passadas. Foi utilizado um algoritmo de Modelo Escondido de Markov para estruturar as rotas de chegadas dos motoristas usando dados de GNSS dos carros. Com o modelo foi possível fazer essa predição em tempo real do destino dos motoristas e otimizar o planejamento das viagens, sugerindo caminhos que evitem muito tráfego.

Como exemplos já aplicados de transporte inteligente em larga escala, existem diversas cidades que disponibilizam dados de ônibus em tempo real para seus habitantes, como é o caso de São Paulo[1], Rio de Janeiro[2] e Santos[3]. Entramos em contato com a SPTrans, companhia que gerencia o transporte público de ônibus na cidade de São Paulo, e conseguimos a informação de que cada veículo é configurado manualmente antes de começar o trajeto e que podem ocorrer falhas humanas, as quais são constantemente monitoradas. Além disso, entramos em contato com uma empresa privada que fornece, para diversas cidades, um sistema de monitoramento de ônibus que é capaz de reportar inconsistências entre a rota que o veículo deveria fazer e o trajeto que está sendo efetuado. O sistema toma como base locais estratégicos de cada rota, usando isso para diferenciá-las. Esse tipo de algoritmo será abordado neste artigo.

O projeto que mais se relaciona com nossa proposta é o do sistema *RioBusData*. Nele, foram investigadas possíveis rotas incorretas entre os ônibus do Rio de Janeiro e foi proposto um algoritmo que utiliza de uma Rede Neural Convolucional para tentar classificar qual rota das 486 estudadas está sendo realizada, e caso haja diferença entre a rota predita pelo algoritmo e a rota configurada, tem-se uma inconsistência. O sistema consegue exibir as inconsistências para o usuário com atraso de 2 horas, que é o tempo que o algoritmo leva para compor uma entrada a ser predita, tornando o monitoramento das rotas mais fácil de ser realizado, porém não em tempo real [Bessa et al. 2016]. A solução aqui proposta se diferencia das demais porque usa aprendizado de máquina para prever rotas, recuperando de inconsistências em tempo real, como pode ser visto na Tabela 1.

Referência	Previsão de Rotas	Recuperação de Falhas	Aprendizado de Máquina	Em Tempo Real
Souza et. al.	✗	✗	✗	✓
Neves et. al.	✗	✓	✓	✓
Miranda et. al.	✗	✗	✗	✓
Saad et. al.	✗	✗	✗	✓
Gong et. al.	✗	✗	✓	✓
Simmons et. al.	✓	✗	✓	✓
Bessa et. al.	✓	✓	✓	✗
Este Artigo	✓	✓	✓	✓

Tabela 1. Resumo comparativo dos trabalhos relacionados.

[1]http://olhovivo.sptrans.com.br/
[2]http://luis.impa.br/riobas/
[3]https://quantotempofalta.piracicabana.com.br/

3. Fundamentação

Esta seção apresenta o modelo do sistema Circulino, usado como estudo de caso neste trabalho, e descreve o problema que existe atualmente nesse sistema. Antes disso, porém, é preciso definir os conceitos de rota, trajeto e linha para este trabalho: rota é o conjunto de coordenadas geográficas previamente definidas que devem ser percorridas em determinada ordem; trajeto é o conjunto de coordenadas geográficas percorridas sequencialmente por um veículo ao seguir uma rota; linha é o nome dado para se referir a uma rota.

O serviço de transporte da Unicamp conhecido como *Veículo Circular Interno* é composto por ônibus que fazem 4 diferentes linhas dentro da universidade, são elas: Circular 1, Circular 2 Museu, Circular 2 FEC e Circular Noturno[4] . Para melhorar esse serviço, a iniciativa *Smart Campus*[5] desenvolveu o projeto do *Circulino* [Barbosa et al. 2019], que disponibiliza para os usuários a posição do veículo em tempo real por meio do aplicativo Unicamp Serviços e de uma página *web*. A solução inclui um dispositivo IoT composto por um módulo GSM GPS, um sensor de temperatura, um microcontrolador e botões de configuração. Com base nos dados dos sensores é construída uma estrutura JSON composta por campos que incluem latitude, longitude e *timestamp* do momento da coleta. Esses dados são enviados para a plataforma em nuvem da Konker[6] que, por sua vez, interage com as interfaces que disponibilizam as informações para os usuários.

Na concepção do *Circulino*, o motorista do ônibus deve apertar um botão no aparelho que contém o GPS para identificar qual a rota que ele realizará. Eventualmente, o motorista aciona o botão errado ou simplesmente esquece desse passo, gerando uma inconsistência que reflete na informação apresentada ao usuário. Assim sendo, o usuário veria pelo aplicativo um ônibus fora da rota esperada, dificultando seu planejamento para embarcar no veículo correto. Se fosse possível identificar instantaneamente esse tipo de erro, sua recuperação seria rápida, garantindo a qualidade do serviço.

4. Metodologia

A partir da investigação da rota dos circulares, foi possível perceber que todos apresentam o mesmo ponto inicial, o que torna o problema ainda mais difícil de se resolver se quisermos que o sistema alerte inconsistências logo no momento que o ônibus sai do terminal. Porém, como cada linha tem uma grade horária específica, o mapeamento do tempo com a posição pode facilitar a análise.

Sabendo disso, a solução encontrada envolveu a criação de algoritmos que predizem em tempo real qual rota o ônibus está realizando com base em suas leituras, assim como foi feito no *RioBusData* [Bessa et al. 2016]. A vantagem de usar essa metodologia é poder sugerir a rota correta caso haja divergências com relação a configuração manual, facilitando a recuperação do erro. Assim sendo, para conseguir desenvolver esses algoritmos, primeiro foi necessário construir uma base de dados de trajetos (Seção 4.1), depois elaborar algoritmos de predição (Seção 4.2), e por último realizar o treino e testes dos modelos desenvolvidos (Seção 4.3).

[4]https://www.prefeitura.unicamp.br/servicos/diretoria-de-servicos-de-transporte/mapa-circulares
[5]https://smartcampus.prefeitura.unicamp.br/
[6]http://www.konkerlabs.com.br/

4.1. Construindo a Base de Dados de Trajetos

Primeiramente, foi realizado um pré-processamento dos dados coletados no período de 11 de Junho de 2018 até 18 de Abril de 2019 de modo a agrupar as entidades em trajetos que percorreram rotas completas, reunindo leituras enviadas em sequência e ininterruptamente por um determinado dispositivo em que somente a primeira e a última leitura se encontram próximas em relação ao ponto do terminal de ônibus.

O algoritmo de pré-processamento (Algoritmo 1) levou em conta que as entidades já se encontravam agrupadas por dispositivo e ordenadas de acordo com o horário da coleta no banco de dados. Ao final, é esperado que o trajeto gerado comece em um raio de até 100 metros de proximidade do terminal, se afaste dele ao realizar o caminho e termine no mesmo raio de proximidade dele. O valor 100 foi escolhido para englobar toda a região da rua em que os ônibus costumam ficar estacionados em fila, de modo que um trajeto não comece enquanto o veículo se mantiver nessa rua.

Algorithm 1 Pré-processamento de dados para geração de trajetos.

```
1:  function CRIATRAJETOS(baseDeDados[])
2:      trajetos ← [ ]; estado ← iniciando
3:      for i ← 1 to length(baseDeDados) do
4:          if estado == iniciando then
5:              novoTrajeto ← [ ]
6:              if distanciaEntre(baseDeDados[i], terminalDeOnibus) > 100 then
7:                  estado ← percorrendo; novoTrajeto.append(leitura)
8:          else if estado == percorrendo then
9:              novoTrajeto.append(leitura)
10:             if distanciaEntre(baseDeDados[i], terminalDeOnibus) < 100 then
11:                 estado ← iniciando; trajetos.append(novoTrajeto)
        return trajetos
```

Um exemplo de um trajeto gerado por esse pré-processamento pode ser visto na Figura 1, onde é possível ver que os pontos referenciam a rota do *Circular 2 Museu*, em que o ônibus inicia seu trajeto no ponto mais violeta e termina no ponto mais avermelhado passando por todo espectro de cores. Uma vez tendo todos os trajetos bem definidos, foi possível obter um arquivo CSV (*Comma Separated Values*) com os campos de latitude, longitude, identificador da linha, data e hora de coleta e índice do trajeto, em que leituras que tenham o mesmo índice do trajeto pertencem ao mesmo trajeto, e o identificador da linha representa qual informação da rota foi configurada pelo motorista.

Levando em conta que alguns algoritmos de aprendizado de máquina exigem uma entrada de tamanho fixo, ou seja, a quantidade de *features* de entrada deve ser igual para todos os trajetos testados, e que cada trajeto tem um número diferente de leituras, foram aplicados dois novos pré-processamentos dos dados para produzir essa normalização. O primeiro é chamado de subamostragem e o método utilizado consiste em transformar todas as leituras dentro de uma janela de tempo em uma única, correspondendo a média de latitude, longitude, e hora entre elas. No caso, foi escolhida uma janela de dez segundos e um passo também de dez segundos porque esse é o período estimado de uma parada do veículo em um ponto de ônibus. Tomando como base que um trajeto do circular

demora aproximadamente 30 minutos para ser concluído, então após aplicar o primeiro pré-processamento espera-se obter trajetos de quantidade de leituras próximas de 180, a partir disso, o segundo pré-processamento foi proposto: a discretização do comprimento total do trajeto em 60 partes de iguais comprimentos e obtendo dessa forma o valor médio das variáveis estudadas para cada uma dessas regiões. O número 60 foi escolhido porque haverá em torno de 3 leituras agrupadas por cada grupo aglutinado, o que não representa grande perda de informação e minimiza o uso de dados e tempo de processamento de treinamento dos algoritmos. Ao final, todos os trajetos terão exatamente 60 leituras, permitindo o treinamento em algoritmos que exigem entrada fixa.

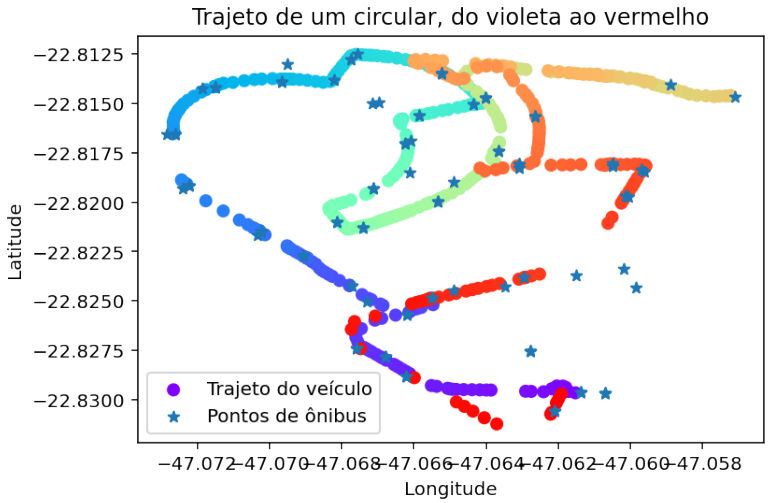


Figura 1. Exemplo de trajeto do circular gerado pelo Algoritmo 1.

Tendo em mente que o algoritmo buscado precisa ter capacidade de inferência da rota do ônibus em tempo real durante o trajeto e não apenas no seu término, foi necessário aumentar a base de dados de trajetos de modo a simular subtrajetos, que são definidos como sendo o conjunto das i primeiras leituras de um trajeto. Assim sendo, dado um trajeto T completo de n leituras, criou-se um novo subtrajeto apenas com a primeira leitura de T, outro apenas com a primeira e segunda leituras de T, e assim sucessivamente até obter n-1 novos subtrajetos, o que simularia T sendo percorrido. No caso de um subtrajeto que tenha uma quantidade de leituras menor que 60, ele pode ser facilmente transformado em uma entrada de 60 leituras simplesmente adicionando zeros nos parâmetros faltantes, mantendo assim uma entrada fixa.

4.2. Algoritmos Utilizados

Foram propostos e testados ao total sete algoritmos, entre os quais um é programado explicitamente e seis são de aprendizado de máquina. O algoritmo programado explicitamente foi chamado de *Algoritmo de Georreferência* e compara cada uma das coordenadas de um dado trajeto com pontos que são exclusivos para cada rota. Uma vez tendo os pontos definidos, foi possível propor o Algoritmo 2, em que os valores usados para as distâncias foram obtidos com base em uma estimativa da proximidade que um circular deveria passar por um ponto para ser considerado de uma determinada rota. Desse modo, é possível por eliminação determinar qual é a rota que o ônibus está realizando.

Algorithm 2 Georreferência.

```
1: function TESTAALGORITMOGEORREFERÊNCIA(trajeto[])
2:     if menorDistanciaEntre(trajeto, pontoCircularMuseu) < 300 then
3:         return circularMuseu
4:     else if menorDistanciaEntre(trajeto, pontoCircular1) < 250 then
5:         return circular1
6:     else if menorDistanciaEntre(trajeto, pontoCircularFec) < 250 then
7:         return circularFec
8:     else
9:         return circularNoturno
```

Para os algoritmos de Aprendizado de Máquina, foram investigados a Rede Neural do *RioBusData* [Bessa et al. 2016], uma Rede Neural Convolucional voltada para dados unidimensionais, *Support Vector Machine* (SVM), Floresta Aleatória, *Naïve Bayes* e um conjunto de modelos de Séries Temporais. O algoritmo de Séries Temporais consistiu em treinar um modelo para cada rota predefinida, de modo que para uma determinada rota A, uma base de dados com latitude, longitude e tempo de cada leitura de todos os n trajetos de A, que vão de A_1 até A_n seja criada com o intuito de concatenar a última leitura de A_i com a primeira de A_{i+1}, para todo $1 \leq i < n$. Uma vez treinados os modelos para A e para todas as outras rotas, é possível utilizá-los para classificar rotas da seguinte forma: dada uma nova leitura l_i, cada modelo de série temporal deverá prever quais seriam os valores de latitude, longitude e tempo da leitura subsequente l_{i+1}, e então quando l_{i+1} for lido, a distância aritmética entre l_{i+1} e a predita por cada modelo é calculada, e a rota correspondente ao modelo que mais se aproximou da leitura correta é escolhida pelo algoritmo, conforme pode ser visto no Algoritmo 3.

Algorithm 3 Séries Temporais

```
1: function TESTASERIESTEMPORAIS(leituraAnterior, leituraAtual, modelos[])
2:     distancias ← []
3:     for modelo ∈ modelos do
4:         leituraPredita ← modelo.predizer(leituraAnterior)
5:         distancias.append(distanciaEntre(leituraPredita, leituraAtual))
6:     return menorCircularEntre(distancias)
```

4.3. Treinos e Testes dos Algoritmos

Uma vez tendo a base de dados e os algoritmos escolhidos, todos os trajetos e subtrajetos foram então separados em 80% para treino e 20% para teste. Para os algoritmos de Floresta Aleatória e Rede Neural dividiu-se a base de treino em 80% para treino de parâmetros e 20% para validação cruzada, de modo a encontrar os melhores parâmetros maximizando a acurácia, que, para este trabalho, é definida como a razão entre a quantidade de predições certas e a quantidade total de predições. Uma vez tendo os melhores parâmetros, os resultados foram testados na base de dados de teste e comparados com seus identificadores de linha correspondentes. No intuito de medir a eficácia dos algoritmos para cada trajeto sendo percorrido, definimos o conceito de porcentagem de completude como sendo a razão entre a quantidade de leituras do subtrajeto e seu respectivo trajeto

principal, e com base nela avaliar a acurácia dos modelos entre zero a 100 de porcentagem de completude dos trajetos. O melhor algoritmo é aquele que tiver a melhor acurácia somando cada porcentagem de completude, o que matematicamente é proporcional a área do gráfico de acurácia versus porcentagem de completude para cada curva de algoritmo.

Com o propósito de mostrar que a janela de 60 é a escolha mais apropriada, o melhor algoritmo da etapa anterior foi selecionado e com ele foram gerados diferentes modelos do mesmo algoritmo utilizando diferentes tamanhos de janelas. Foram testadas janelas de 5, 15, 30, 45, 60, 90, 120 e 180 leituras, e cada novo modelo foi testado em sua acurácia em relação a sua porcentagem de completude do trajeto, de modo a maximizar a acurácia e minimizar o tamanho da janela, dando sempre preferência ao primeiro.

5. Resultados e Análise

Um gráfico comparativo com os resultados de cada algoritmo pode ser visto na Figura 2a, e a área de cada curva pode ser vista na Tabela 2. Percebe-se que o algoritmo de Floresta Aleatória foi o que obteve o melhor resultado, tendo a maior área de cobertura, alcançando uma média de 96% de acurácia para os trajetos completos, e ficando acima de todos os outros na maior parte do alcance do gráfico. Esse resultado se deve ao fato de que esse algoritmo é capaz de analisar independentemente cada parâmetro de entrada, conseguindo lidar muito bem com subtrajetos que foram preenchidos com zeros nos parâmetros faltantes.

Alg.	Floresta Aleatória	RNC Unid.	Séries Temp.	SVM	RNC do RioBus	Naïve Bayes	Alg. Geo.
Área	7815,32	7092,80	6887,24	6370,78	4536,22	4280,05	3867,32

Tabela 2. Área da curva de cada algoritmo trabalhado.

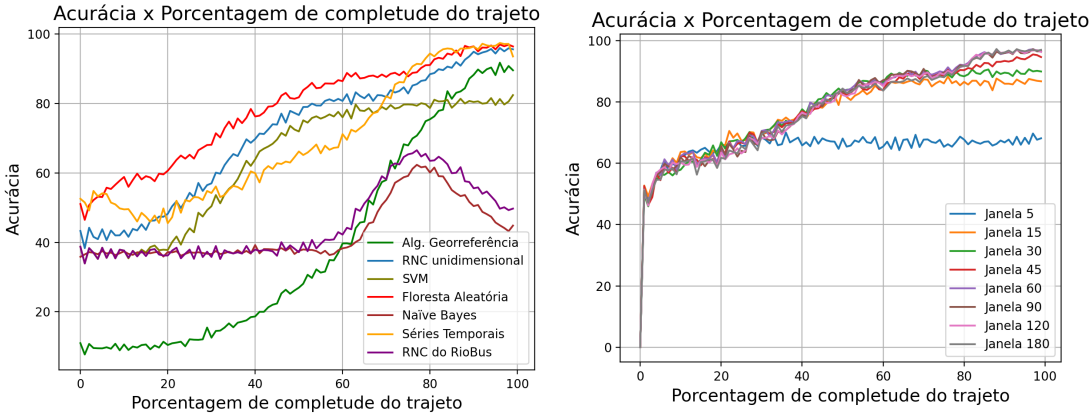
É esperado que o algoritmo de Georreferência, que é o único que não utiliza a informação de tempo, tenha um início inferior aos demais, porque é impossível diferenciar cada trajeto logo no começo apenas pela posição. Ao final, esse algoritmo conseguiu alcançar 90% de acurácia, já que um trajeto completo tende a ter todas as informações necessárias para se realizar a predição correta. Nota-se que como o algoritmo utilizado pela empresa contatada também não utiliza a informação de tempo, tem-se uma grande evidência de que ele teria uma acurácia menor do que os demais dentro do contexto da Universidade, sendo necessário um novo estudo para confirmar se o mesmo se aplica no contexto das cidades.

A Rede Neural do *RioBusData* e o *Naïve Bayes* apresentaram quedas bruscas nos 20% finais de completude do trajeto, o que não é esperado, pois uma porcentagem de completude maior implica em mais dados para serem analisados, facilitando a predição. As previsões equivocadas dos algoritmos podem ser explicadas pela imprecisão dos valores do GPS, que podem variar por metros, e pela presença de trajetos com configurações erradas no banco de dados, atrapalhando o treino e a checagem dos resultados.

Para os testes do melhor tamanho de janela, foi escolhido como modelo o algoritmo de Floresta Aleatória. Os resultados obtidos podem ser vistos na Figura 2b e mostram que a janela de 60 realmente é a melhor escolha, pois maximiza a acurácia,

equiparando-se as janelas de tamanhos até 180, sendo a mínima janela necessária que maximiza o acerto.

Uma vez tendo escolhido o melhor algoritmo, é necessário realizar as seguintes perguntas: 1) o algoritmo é capaz de substituir totalmente a configuração manual da rota que está sendo realizada pelo ônibus? 2) o quanto podemos confiar no algoritmo para alertar que há uma inconsistência e recomendar a rota correta? Para a primeira pergunta, seria preciso determinar qual a acurácia de se configurar manualmente a rota, e caso a acurácia do melhor algoritmo para uma determinada faixa de porcentagem de completude seja maior do que a manual, pode-se substituir; para a segunda pergunta, é fundamental traçar um limiar de acurácia mínima e seguir os seguintes passos: caso a predição do melhor algoritmo seja diferente da configuração manual e o valor da acurácia do algoritmo na atual porcentagem de completude do trajeto seja maior do que a acurácia mínima, alerte inconsistência e sugira a rota correta. A acurácia da configuração manual pode ser estimada analisando manualmente dentre um grupo trajetos quais não condizem com a rota configurada, e a acurácia mínima é definida arbitrariamente pelos gerenciadores do Sistema, dependendo se querem um sistema que alerte inconsistência apenas quando tem certeza ou com o mínimo de confiança. Ambas serão abordadas e definidas em trabalhos futuros.



(a) Comparação de algoritmos trabalhados.

(b) Comparação das diferentes janelas testadas.

Figura 2. Gráficos comparativos de algoritmos e teste das janelas.

6. Conclusão e Trabalhos Futuros

Este artigo propôs algoritmos que podem ser utilizados para identificar em tempo real que o trajeto realizado por um veículo difere da rota esperada. Além disso, foi mostrado que o modelo de análise proposto apresenta melhor acurácia do que a solução de referência, que observamos que é utilizada por uma empresa que atua em várias cidades brasileiras.

O próximo passo é definir as acurácias mínima e da configuração manual do sistema Circulino, bem como estudar novas opções de modelos para conseguir melhorar ainda mais a acurácia do melhor algoritmo, podendo alertar inconsistências do sistema com maior antecedência. Adicionalmente, serão investigadas outras bases de dados de veículos que possuam rotas definidas para determinar se as soluções propostas na universidade também poderiam ser aplicadas em outros contextos, propondo melhorias nos sistemas das cidades, por exemplo.

Referências

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.

Barbosa, R. A., Sousa, R. P., Oliveira, F. A., Oliveira, H. C., Luz, P. D. G., and Manera, L. T. (2019). Circulino: An iot solution applied in the university transport service. In Iano, Y., Arthur, R., Saotome, O., Vieira Estrela, V., and Loschi, H. J., editors, *Proceedings of the 4th Brazilian Technology Symposium (BTSym'18)*, pages 503–514, Cham. Springer International Publishing.

Bessa, A., de Mesentier Silva, F., Nogueira, R. F., Bertini, E., and Freire, J. (2016). Riobusdata: Outlier detection in bus routes of rio de janeiro. *CoRR*, abs/1601.06128.

Chen, C., Zhang, D., Li, N., and Zhou, Z. (2014). B-planner: Planning bidirectional night bus routes using large-scale taxi gps traces. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1451–1465.

Gong, J., Liu, M., and Zhang, S. (2013). Hybrid dynamic prediction model of bus arrival time based on weighted of historical and real-time gps data. In *2013 25th Chinese Control and Decision Conference (CCDC)*, pages 972–976.

IBGE. População rural e urbana. *IBGEeduca*.

Martins, K. S. and Cunha, F. D. (2018). Explorando dados urbanos: um estudo usando viagens de tá xi da cidade de s ão francisco. In *Anais do II Workshop de Computação Urbana*, Porto Alegre, RS, Brasil. SBC.

Miranda, W., Mendonça, R., Silva, A., Curvello, A., Souza, F., and Silva, H. (2017). Busme: Automatic bus localization system and route registration. *Procedia Computer Science*, 109:1098–1103.

Neves, D. V. (2019). Uso de aprendizado supervisionado para análise de confiabilidade de dados de crowdsourcing sobre posicionamento de ônibus. In *Dissertação de Mestrado em Sistemas de Informação*.

Nunes, D. E., Fagundes, A., and Mota, V. F. S. (2018). Classificação da qualidade de vias urbanas baseado em sensoriamento participativo. In *Anais do II Workshop de Computação Urbana*, Porto Alegre, RS, Brasil. SBC.

Saad, S. A., Hisham, A. . B., Ishak, M. H. I., Fauzi, M. H. M., Baharudin, M. A., and Idris, N. H. (2018). Real-time on-campus public transportation monitoring system. In *2018 IEEE 14th International Colloquium on Signal Processing Its Applications (CSPA)*, pages 215–220.

Simmons, R., Browning, B., Yilu Zhang, and Sadekar, V. (2006). Learning to predict driver route and destination intent. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 127–132.

Sousa, P., Costa, J., Castro, A., Neto, W., Bezerra, C. I. M., and Coutinho, E. (2019). Uma infraestrutura para o monitoramento e predição de rotas e paradas de Ônibus no transporte universitário.

Viana, J. D. F., Braga, O., Silva, L., and Neto, F. M. (2019). Analyzing patterns of a bicycle sharing system for generating rental flow predictive models. In *Anais do III Workshop de Computação Urbana*, pages 57–70, Porto Alegre, RS, Brasil. SBC.