UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JOSÉ LUCIVAN BATISTA FREIRES

UMA APLICAÇÃO PARA O ENRIQUECIMENTO SEMÂNTICO DE TRAJETÓRIAS USANDO ALGORITMOS DE DETECÇÃO DE PARADAS

QUIXADÁ

2018

JOSÉ LUCIVAN BATISTA FREIRES

UMA APLICAÇÃO PARA O ENRIQUECIMENTO SEMÂNTICO DE TRAJETÓRIAS USANDO ALGORITMOS DE DETECÇÃO DE PARADAS

Monografia apresentada no curso de Sistemas de Informação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Sistemas de Informação. Área de concentração: Computação.

Orientadora: Dra. Ticiana Linhares Coelho da Silva

QUIXADÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

F933a Freires, José Lucivan Batista.
Uma aplicação para o enriquecimento semântico de trajetórias usando algoritmos de detecção de paradas / José Lucivan Batista Freires. – 2018.
60 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2018.
Orientação: Profa. Dra. Ticiana Linhares Coelho da Silva .

1. Semântica. 2. Trajetória. 3. Correspondência de mapas. 4. Ponto de interesse. I. Título.
CDD 005

JOSÉ LUCIVAN BATISTA FREIRES

UMA APLICAÇÃO PARA O ENRIQUECIMENTO SEMÂNTICO DE TRAJETÓRIAS USANDO ALGORITMOS DE DETECÇÃO DE PARADAS

Monografia apresentada no curso de Sistemas de Informação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Sistemas de Informação. Área de concentração: Computação.

Aprovada em: __/__/____

BANCA EXAMINADORA

Dra. Ticiana Linhares Coelho da Silva (Orientadora)
Universidade Federal do Ceará – UFC

Dr. Regis Pires Magalhães
Universidade Federal do Ceará - UFC

Ma. Lívia Almada Cruz Rafael
Universidade Federal do Ceará - UFC

Este trabalho é dedicado aos meus pais, Aurilene e Luciano, ao meu irmão Yago, a minha avó materna Maria, e ao meu avô materno que já descansa na casa do Senhor.

AGRADECIMENTOS

Agradeço a Deus, por me manter nesse caminho e me ajudar nos desafios que a vida me deu.

Agradeço aos meus pais, pelos sacrifícios e por me darem ajuda, apoio e terem me incentivado a estudar.

Agradeço a minha avó materna, por sempre ter me enchido de amor e carinho em todos os momentos, por sempre acreditar em mim.

Agradeço ao meu avô paterno, por sempre me amar e por ser o exemplo da minha vida.

Agradeço a Profa. Dra. Ticiana Linhares Coelho da Silva, pela excelente orientação, pela paciência, pelos conselhos e pela dedicação.

Agradeço ao Prof. Dr. Davi Romero por ter me aceito como bolsista do PET e pelos excelentes conselhos que contribuíram para a minha formação profissional e acadêmica.

Agradeço ao Prof. Dr. Regis Pires Magalhães e a Profa. Ma. Lívia Almada Cruz Rafael, por participarem da banca examinadora e pelos valiosos conselhos e sugestões.

Agradeço aos demais professores pelo vasto conhecimento que me foi passado.

Agradeço, em especial, aos meus amigos Naélio, Rodrigo, Fagner, João Mateus, Gabriel, Patrick, Roy e Andreazo, por terem me ajudaram em momentos difíceis e estiveram comigo desde o começo da graduação.

Agradeço aos meus outros amigos da turma de graduação a qual faço parte e do grupo de bolsista do PET, que contribuíram direta ou indiretamente na minha jornada.

"A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo."

(Albert Einstein)

RESUMO

Este trabalho descreve um processo e uma aplicação para o enriquecimento semântico de trajetórias, por meio de trajetórias de taxistas. O processo consiste na identificação dos pontos onde o carro está parado e em etapas para remoção de inconsistência dos dados. Os dados aqui utilizados, pertencem ao T-Drive na cidade de Pequim na China. Neste trabalho, para a remoção de dados com inconsistências, é utilizado a correspondência de mapas. Aqui, são analisados e utilizados os algoritmos IB-SMoT e CB-SMoT para realizarem o enriquecimento semântico das trajetórias. Para um bom desempenho e para a obtenção de bons resultados, é realizado uma análise dos parâmetros dos algoritmos de enriquecimento semântico. É apresentando uma API para a coleta de pontos de interesse da região escolhida. A aplicação desenvolvida aqui, utiliza desses dois algoritmos para realizar o enriquecimento semântico de forma que um usuário leigo consiga utilizar.

Palavras-chave: Enriquecimento semântico. Trajetória. Correspondência de mapas. Detecção de paradas. Ponto de Interesse.

ABSTRACT

This paper describes a process and an application for the semantic enrichment of trajectories, through trajectories of taxi drivers. The process consists in identifying the points where the car is stopped and in steps to remove data inconsistency. The data used here, belong to T-Drive in the city of Beijing in China. In this paper, for the removal of data with inconsistencies, map matching is used. Here, the IB-SMoT and CB-SMoT algorithms are analyzed and used to perform the semantic enrichment of the trajectories. For a good performance and to obtain good results, an analysis of the parameters of semantic enrichment algorithms is performed. It is presenting an API for collecting points of interest from the chosen region. The application developed here, uses these two algorithms to peform semantic enrichment in a way that a lay user can use.

Keywords: Semantic enrichment. Trajectory. Map matching. Stops detection. Point of interest.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O Sistema de Posicionamento Global (*Global Positioning System*, GPS) foi criado para fins militares e recebe sinais enviados por satélite, além de determinar onde a pessoa está (PARKINSON, 1996). O GPS é um sistema de navegação por satélite que fornece coordenadas geográficas em tempo real, possibilitando reportar o local onde o dispositivo se encontra na Terra. Por meio dele, é possível gravar o caminho que um objeto em movimento percorreu. É possível encontrar GPS nos sistemas de navegação de carros e outros meios de transporte. Nas últimas décadas, os celulares modernos (*Smartphones*) também começaram a dispor de um GPS integrado, sendo este acessível através de seus próprios aplicativos.

Os dispositivos celulares e os veículos estão cada vez mais avançados e comuns nas vidas das pessoas. Devido ao grande número de veículos e celulares que possuem GPS, é possível gerar uma grande quantidade de dados. Esses dados podem ser usados para analisar as trajetórias que esses objetos realizaram, além de outros fatores, como, por exemplo, velocidade, distância, aceleração e paradas realizadas durante o caminho percorrido. Muitas áreas de conhecimento utilizam os dados providos por GPS para estudos como a análise da migração de aves e o congestionamento do trânsito nas cidades e turismo.

Os dados gerados por GPS são espaço-temporais, ou seja, possuem coordenadas geográficas e tempo. Por não possuírem nenhuma outra informação além das que o GPS gera, torna-se difícil a análise dessas trajetórias devido à complexidade de utilizar um grande volume de informações. Esses dados formam trajetórias, também conhecidas como trajetórias brutas. As trajetórias brutas são sequências de pontos espaciais ordenados pelo momento de ocorrência de cada ponto (SPACCAPIETRA et al., 2008).

Para transformar as trajetórias brutas em trajetórias semânticas, é necessário um pré-processamento para a remoção de inconsistências e *outliers*[1]. Essa instabilidade normalmente ocorre quando o GPS pode não armazenar corretamente as informações com precisão ou por conta da perda do sinal, causando a existência de diversos pontos distantes da rota percorrida. Depois desse pré-processamento, a semântica, ou seja, informações adicionais, é adicionada nas trajetórias brutas. Tal processo é conhecido como enriquecimento semântico. As informações que o enriquecimento semântico irá adicionar aos pontos, são referentes aos conceitos de *stops* e *moves*.

[1] Os *outliers* são dados que se diferenciam de todos os outros. São conhecidos informalmente como pontos fora da curva. Em outras palavras, um *outlier* é um valor que foge da normalidade e que pode (e provavelmente irá) causar anomalias nos resultados obtidos por meio de algoritmos e sistemas de análise.

Usualmente, os dados de uma trajetória são representados como uma sequência de pontos no seguinte formato: (*id, latitude, longitude, t*). *Id* representa o identificador do objeto em movimento, *latitude* e *longitude* representam as coordenadas geográficas gravadas pelo GPS, enquanto *t* representa o instante de tempo em que o objeto estava em determinada posição. Uma sequência de pontos representa uma trajetória bruta, não sendo fácil interpretá-la. Desta maneira, não é possível identificar padrões ou características da trajetória. Para que seja possível uma melhor compreensão da trajetória, é necessário realizar um processamento desses pontos e seu enriquecimento semântico (ALVARES et al., 2007a).

Spaccapietra et al. (2008) mostram uma análise do comportamento de cegonhas durante o período de migração. Essa análise serve de exemplo para a compreensão de trajetórias semânticas. Equipadas com um pequeno transmissor, é possível armazenar a sua posição espaço-temporal em intervalos regulares. Uma forma de dar semântica a essas trajetórias é estabelecendo o motivo do pássaro ter parado em um determinado local, como, por exemplo: descansar ou alimentar-se. Outra forma é saber o tempo de permanência no local, conhecer informações sobre as condições do ambiente da trajetória, como: direção do vento, temperatura, objetos naturais e artificiais, e condições do tempo, como chuva, sol, neblina, etc. Essas informações permitem analisar e transformar os dados dessas trajetórias em trajetórias enriquecidas de informações, conhecidas como trajetórias semânticas.

Em Yan et al. (2011a) é apresentado um *framework* que busca dar um contexto para as trajetórias através de pontos, linhas e regiões de interesse para diversos tipos de objetos em movimento. O foco do trabalho de Yan et al. (2011a) está na anotação semântica, que consiste em um pós-processamento das paradas e dos movimentos das trajetórias nos pontos, linhas e regiões de interesse. Em outras palavras, após o processamento das paradas (*stops*) e dos movimentos (*moves*), existe um processo que utiliza essas informações, além das regiões de interesse, para enriquecer as trajetórias semanticamente.

Neste trabalho, utiliza-se de pré-processamento dos dados, processamento de detecção dos *stops* e de enriquecimento semântico. Para isso, este trabalho utiliza um algoritmo que implementa a técnica de *Map Matching* e também algoritmos de detecção de *stops* e *moves* como o IB-SMoT, definido em Alvares et al. (2007a), e o CB-SMoT, definido inicialmente em Palma et al. (2008) e o algoritmo foi ajustado posteriormente em Palma (2008).

A aplicação aqui proposta, busca fazer com que seja possível enriquecer semanticamente trajetórias de objetos em movimento, ou seja, transformar dados

espaço-temporais em uma saída com contexto. Após o término, o usuário poderá visualizar os resultados do enriquecimento semântico. Essas trajetórias semânticas podem ser utilizadas como informações para a tomada de decisões sobre os pontos de interesses.

Pretende-se descobrir, através da saída gerada pela aplicação, os diversos pontos de interesses visitados pelos objetos em movimento. A partir de bases de dados, como a base do TDrive apresentada em Zheng (2011), é possível analisar, por exemplo, os pontos turísticos mais visitados por taxistas durante o seu expediente, ou os restaurantes mais visitados, ou até mesmo, onde os passageiros pedem para que eles reduzam a velocidade para admirarem certos pontos.

Os dados manipulados por este trabalho são históricos, ou seja, em algum outro momento eles foram armazenados. Os resultados deste trabalho podem ser adaptados para a utilização na recomendação de pontos de interesse e reconhecimento dos locais mais visitados pelos taxistas. Através de consultas nos resultados do enriquecimento semântico, é possível descobrir os horários e os locais mais visitados. Além disso, a aplicação pode ser adaptada para a utilização da administração de municípios para a redução do congestionamento em determinadas ruas da cidade. É possível reduzir o congestionamento nas ruas usando os resultados do algoritmo CB-SMoT, pois ele utiliza da velocidade. Uma estratégia pode ser proposta para a redução do congestionamento ao saber quais os locais que mais congestionam.

Este trabalho está organizado da seguinte forma: No Capítulo 2, é realizada uma explicação do problema e de definições relacionadas ao problema em questão. No Capítulo 3, são apresentados o objetivo geral e os objetivos específicos. No Capítulo 4, são mostrados os principais conceitos utilizados no trabalho. No Capítulo 5, são descritos alguns dos trabalhos relacionados e apresentadas as semelhanças e diferenças com este trabalho. No Capítulo 6, os procedimentos metodológicos são definidos. No Capítulo 7, os resultados alcançados são mostrados. No Capítulo 8, são apresentadas as considerações finais e trabalhos futuros.

2 DEFINIÇÃO DO PROBLEMA

Neste Capítulo, são descritas as definições para uma boa formalização deste trabalho e como essas definições se encaixam na problemática. Abaixo, encontram-se as definições formais que foram baseadas nos trabalhos de Alvares et al. (2007a), Alvares et al. (2007b) e Yan et al. (2011a).

Definição 2.0.1. Ponto da Trajetória é um **ponto** p = (id, latitude, longitude, *timestamp*), onde: **id** representa um identificador único do objeto em movimento; **latitude** é a distância em graus de qualquer ponto da Terra em relação à linha do equador; **longitude** é a distância em graus de qualquer ponto da Terra em relação ao Meridiano de Greenwich; e ***timestamp*** é o instante do tempo daquele ponto na determinada posição.

Definição 2.0.2. Trajetória Bruta Sequência gravada de pontos geográficos de um determinado objeto, por exemplo, T = $\{(id, x_0, y_0, t_0), (id, x_1, y_1, t_1), ..., (id, x_n, y_n, t_n)\}$, podendo ser representado como T = $\{P_1, ..., P_n\}$. A sequência é ordenada com base no atributo *timestamp* dos pontos e para os pontos pertencerem a uma mesma trajetória é necessário que possuam o mesmo identificador do objeto.

Definição 2.0.3. Ponto de Interesse Um ponto de interesse (*Point of Interest*, POI) é um objeto geográfico que é interessante para uma aplicação específica, normalmente associada a uma atividade humana. Formalmente, um ponto de interesse é definido como POI = (c, r, l), onde c representa o ponto, r é a área espacial representando a extensão do objeto e l é o rótulo da forma *CAT:N*, onde *CAT* é a categoria do ponto de interesse e *N* é o nome do ponto de interesse. Por exemplo, em um estádio de futebol, a representação do ponto espacial c é o centro do campo de futebol, r é área do raio do estádio, e o rótulo pode ser denominado como estádio para a categoria e Castelão para o nome do ponto de interesse.

Definição 2.0.4. Candidato a *Stop* É uma tupla representada por (R_C, Δ_C), onde: R_C é a geometria do determinado ponto de interesse; e Δ_C é tempo mínimo de duração que um objeto deverá permanecer dentro da área desse determinado ponto de interesse para que possa ser considerado um *stop*.

Definição 2.0.5. *Stop* Dado uma trajetória T e seja A = {C_1 = (R_{C_1}, Δ_{C_1}), C_2 = (R_{C_2}, Δ_{C_2}), ..., C_n = (R_{C_n}, Δ_{C_n})} os candidatos a *stops*. Supondo uma sub-trajetória $< (x_i, y_i, t_i), (x_{i+1}, y_{i+1}, t_{i+1}), ..., (x_{i+l}, y_{i+l}, t_{i+l}) >$ de T, onde existe um (R_{C_k}, Δ_{C_k}) em A de tal forma que $\forall j \in [i, i+l] : (x_j, y_j) \in$

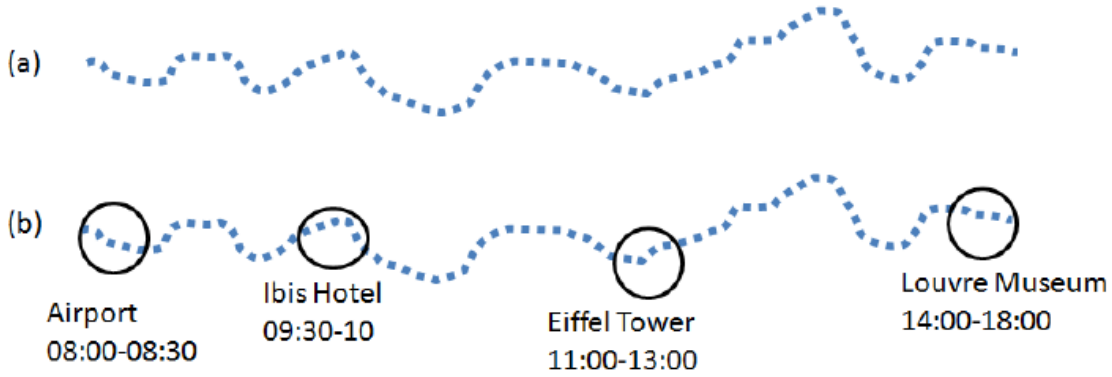
R_{C_k} e $|t_{i+l} - t_i| \geq \Delta_{C_k}$, então a tupla ($R_{C_k}$, t_i, t_{i+l}) é considerada como sendo um *stop* de T em relação aos candidatos em A.

Definição 2.0.6. ***Move*** Dada uma trajetória T e seja A os candidatos a *stops*, um *move* é: (i) uma sub-trajetória de T entre duas paradas temporalmente consecutivas de T; ou (ii) uma sub-trajetória de T entre o ponto inicial de T e o primeiro *stop* de T; ou (iii) uma sub-trajetória de T entre o último *stop* de T e o último ponto de T; ou (iv) a trajetória T por completa, caso T não possua *stops*.

Definição 2.0.7. Trajetória Semântica Dado uma trajetória T, onde para cada ponto existente nessa trajetória é complementado usando anotações. Seja ST a trajetória semântica, ST = {P'_1, P'_2, ..., P'_n}, onde $\forall i \in \{1, 2, ..., n\}$ $P'_i = (id, x, y, t, S)$, onde S representa uma anotação semântica, como, por exemplo, *stop*.

A Figura 1 exemplifica duas trajetórias, onde (a) representa uma trajetória inicial bruta sem semântica, gravada pelo GPS e sem anotação semântica sobre ela e (b) representa a mesma trajetória após o processamento, usando algoritmos em uma aplicação referente ao turismo, destacando-se pontos de interesses desta trajetória. Em (b) identifica-se que entre os candidatos a *stops* são: *Airport, Ibis Hotel, Eiffel Tower* e *Louvre Museum*. Os candidatos a *stops* identificados em (b) foram classificados como *stops* e uma anotação foi adicionada, mostrando o horário de permanência do objeto naquele candidato, enriquecendo a trajetória bruta, tornando-a uma trajetória semântica.

Figura 1 – Exemplo de trajetórias: (a) trajetória bruta; (b) trajetória semântica



Fonte: Alvares et al. (2010)

As trajetórias brutas são constituídas de uma sequência de pontos geográficos ordenados. Os pontos de interesses são localizações específicas que alguém pode achar útil ou interessante, por exemplo, restaurantes, locais históricos, pontos turísticos, bares, entre outros. Ele é chama de candidato a *stop* quando é adicionado um tempo mínimo de duração. Os conceitos relacionados a trajetórias e abordados neste trabalho são *stops* e *moves*. Esses conceitos são usados para a transformação das trajetórias brutas em semânticas, para que seja possível utilizar essas trajetórias para decisões futuras.

O problema de que trata este trabalho, consiste em criar um processo para o enriquecimento semântico de trajetórias, a partir de pontos gerados pelo GPS. Para alcançar este objetivo, é necessário identificar os *stops* e enriquecer semanticamente os pontos que pertencem a esses *stops* através dos pontos gerados pelo GPS após um pré-processamento.

3 OBJETIVOS

Neste Capítulo, são definidos o objetivo geral e os objetos específicos a serem alcançados neste trabalho.

3.1 Objetivo Geral

Desenvolver e disponibilizar uma aplicação WEB e um *framework* para enriquecer semanticamente trajetórias utilizando os conceitos de *stops*, *moves* e pontos de interesses para dados históricos.

3.2 Objetivos específicos

Para alcançar o objetivo geral, os seguintes objetivos específicos devem ser alcançados:

a) Reduzir a quantidade de *outliers* e pontos com a posição geográfica errada nos dados de entrada.
b) Experimentar os melhores parâmetros dos algoritmos de *stops* realizando ajustes a fim de identificar *stops* nas trajetórias.
c) Identificar *stops* através de algoritmos de detecção de *stops*.
d) Enriquecer semanticamente os dados utilizando os *stops*.
e) Organizar os *stops* de forma a gerarem informações para decisões.
f) Criar um *framework* para a execução dos procedimentos para enriquecimento semântico.
g) Desenvolver uma aplicação para a execução dos enriquecimento semântico.
h) Avaliar a eficiência e a precisão da estratégia proposta.

4 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo, são abordados os principais conceitos utilizados neste trabalho e como esses conceitos o fundamentam.

4.1 Map Matching

Durante a captura das posições do objeto, muitas vezes não é possível armazenar corretamente a posição do dispositivo que está sendo utilizado, ocasionando trajetórias fora das ruas. *Map Matching* é um processo para a solução deste problema. Ele combina um mapa eletrônico com a localização de informações para obter a posição real dos veículos em uma rede de ruas. Visualmente, é difícil identificar a qual rua uma determinada trajetória pertenceria, mas quando tem-se milhares de pontos em uma base de dados, isso se torna trabalhoso e impossível de se realizar manualmente. O *Map Matching* é a solução do problema de como fazer com que os pontos geográficos armazenados consigam corresponder com um modelo lógico do mundo real, como um sistema de informação geográfica (LIU; LIU, 2007).

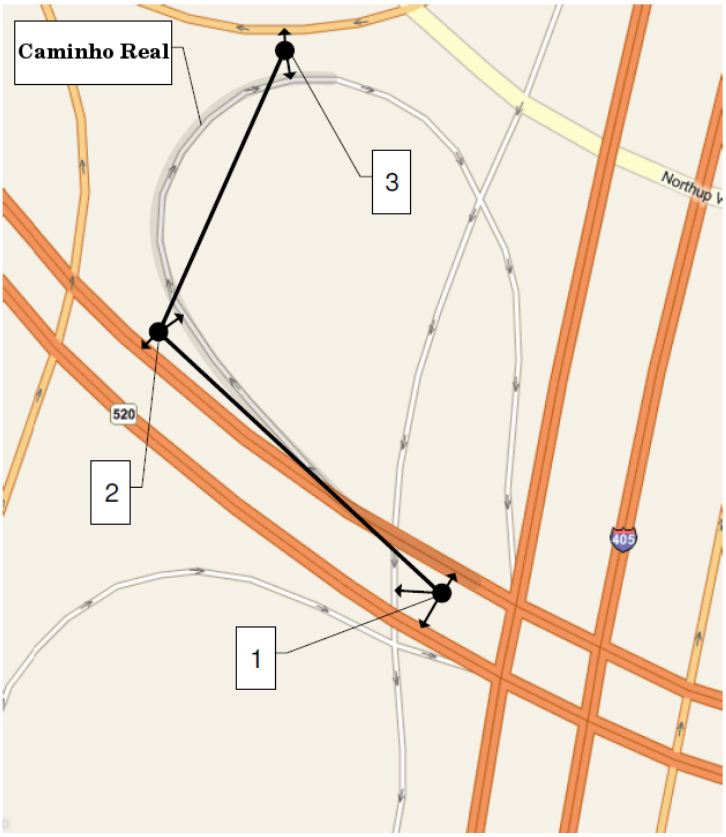
O algoritmo de Liu e Liu (2007) para o *Map Matching* pode ser dividido em três etapas: primeiro, encontrar o caminho que o veículo está percorrendo atualmente; segundo, projetar o ponto atual de posicionamento para o caminho que o veículo está viajando na rede de ruas; terceiro, criação de novos pontos para melhorar a trajetória definida entre dois pontos.

Na Figura 2, pode-se ver uma sequência de pontos fora da rua. Esses pontos foram gravados pelo GPS com erros de precisão, não coincidindo com o caminho real. Normalmente, a abordagem mais comum a ser usada é ter uma sequência de pontos gravados, por exemplo, pelo GPS, e relacioná-los à arestas de um grafo relacionado à rede de ruas existente, que representem a viagem do objeto.

Uma rede de ruas é representada através um grafo, onde os vértices do grafo são pontos na rua, usualmente são interseções de ruas, e a aresta do grafo é a rua em si. A Figura 3 apresenta um exemplo de um grafo bidirecional[1] que busca representar ruas de mão dupla, onde os vértices podem ser pontos específicos de um bairro, como interseções das ruas, enquanto as arestas seriam as ruas. Geralmente, os pontos são relacionados e colocados dentro de uma aresta da rede através de funções de distância, buscando a aresta onde há maior probabilidade do ponto estar localizado baseando na sequência dos pontos.

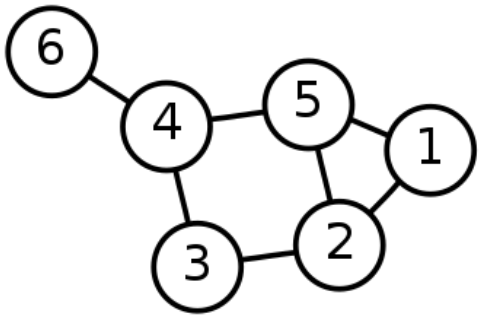
[1] As arestas não possuem sentido

Figura 2 – Exemplo do funcionamento do *map matching*



Fonte: Newson e Krumm (2009)

Figura 3 – Exemplo de um grafo bidirecional para representar a rede de ruas



Fonte: Wikipédia (2018)

Neste trabalho, são utilizados algoritmos para a realização do *map matching*. O algoritmo utilizado para fazer isso, utiliza o *Hidden Markov Model* para encontrar a rota mais provável representada por uma sequência com registro de data e hora de latitude / longitude (NEWSON; KRUMM, 2009).

4.2 Clusterização

A clusterização busca encontrar grupos de objetos, relacionando esses objetos para que os objetos de um mesmo grupo sejam similares uns com os outros. Esses grupos também são denominados *clusters*. Desta forma, os objetos de um mesmo *cluster* devem possuir uma alta similaridade, serem parecidos ou seguirem um padrão, mas eles devem ser dissimilares de objetos de outros *clusters* (TAN; STEINBACH; KUMAR, 2005).

Essa técnica é mais utilizada quando o objetivo é reduzir o número de objetos dentro de um conjunto de dados, dividindo-os em *clusters* com características específicas. O problema desse método se deve ao fato de encontrar a melhor maneira de realizar o agrupamento dos objetos, pois os parâmetros afetam significativamente a criação dos *clusters* (CASSIANO, 2014).

Para encontrar os *clusters* similares entre si, é necessário quantificar a similaridade entre os objetos. As medidas de similaridade são representadas por uma distância entre dois objetos, e esse valor representa como serão formados os grupos. Objetos com distância menor formarão grupos mais similares. Assim, quanto menor a distância entre os objetos, mais similares eles serão (CASSIANO, 2014).

Existem diversas aplicações que utilizam a clusterização, por exemplo: a biologia e a medicina a utilizam para agrupar casos de doenças; o desenvolvimento urbano utiliza para criar clusters geográficos de empresas competitivas; no *marketing* é mais utilizado para identificar grupos distintos de clientes.

No presente trabalho, os *clusters* são utilizados como forma de determinar grupos de pontos geográficos de um veículo, buscando em uma trajetória identificar um *stop* a partir desses *clusters*. Sua definição é utilizada no algoritmo de CB-SMoT.

4.3 DBSCAN

Algoritmos de clusterização são bastante usados para a identificação de classes em bancos espaciais. *Density-based spatial clustering of applications with noise* (DBSCAN) é um algoritmo de clusterização proposto por Ester et al. (1996). Ele utiliza do mínimo de conhecimento sobre o domínio para determinar os parâmetros de entrada e agrupamentos de forma arbitrária, possuindo uma boa eficiência em grandes bases de dados. DBSCAN é baseado na densidade de *clusters*, projetado para descobrir os *clusters* de forma arbitrária. Dado um conjunto de pontos em algum espaço, ele agrupa os pontos que estão mais intimamente próximos,

marcando como *outliers* os pontos que ficam sozinhos em regiões de baixa densidade.

Ester et al. (1996) determinam que o algoritmo DBSCAN pode ser aplicado para espaços Euclidianos de duas e três dimensões, como também, para qualquer característica de espaço de alta dimensionalidade. A distância Euclidiana[2] é a principal função de distância usada, mas é possível escolher outras funções de distâncias dependendo da aplicação. Para cada ponto do *cluster*, a vizinhança em um determinado raio deve conter pelo menos um número mínimo de pontos, isto é, a densidade na vizinhança tem de exceder um certo limite definido como parâmetro. Esse parâmetro que define um limite de pontos é chamado de *MinPts*, enquanto o parâmetro que define um raio para determinar a quantidade de pontos vizinho, é chamado de *Eps*. A forma como a vizinhança irá se formar, será através de uma função de distância escolhida. *MinPts* e *Eps* são os parâmetros necessários utilizados como entrada no algoritmo.

A ideia do método DBSCAN é que, para cada ponto de um *cluster*, a vizinhança para um dado *Eps* contém, no mínimo, certo número de pontos, ou seja, a densidade na vizinhança tem que exceder o *MinPts*. Para entender o método é necessário conhecer algumas definições específicas listadas abaixo (CASSIANO, 2014).

Definição 4.3.1. Vizinhança de um ponto A vizinhança de um ponto P com raio *Eps* é chamado de *Eps*-Vizinhança de P é dado por: $NEps(p) = \{q \text{ em } dist(p,q) < Eps\}$. A função $dist(p,q)$ representa a distância entre o ponto p e o ponto q.

Na Figura 4, NEps(A) são todos os pontos dentro do raio *Eps* a partir de A, 5 pontos e o ponto B. Existem alguns tipos de pontos: *core point, border point* e *noise point*.

Definição 4.3.2. *Core Point* Um ponto será *Core point*, se o número de pontos dentro de uma determinada vizinhança em torno do ponto, conforme determinado pela função de distância e do parâmetro *Eps*, exceda o limite do parâmetro *MinPts*.

Definição 4.3.3. *Border Point* Um ponto será *border point*, se ele não for um *core point*, estiver dentro da vizinhança de um *core point* e não possuir uma quantidade suficiente de *MinPts* dentro do *Eps*.

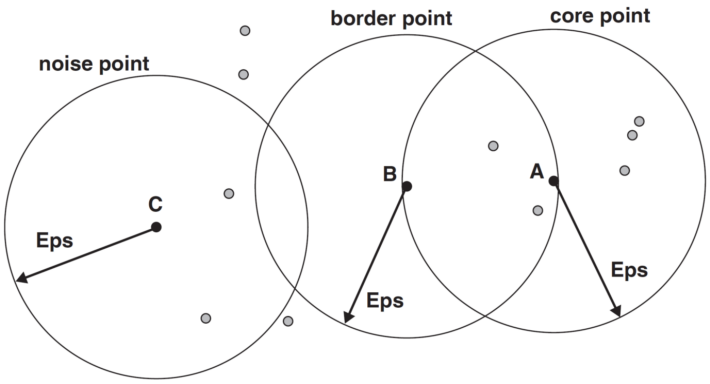
Definição 4.3.4. *Noise Point* Um ponto será *noise point* caso não seja um *core point* ou *border point*. Portanto, um *noise point* não possui *MinPts* suficientes e não está dentro de um *core point*.

[2] Para pontos bidimensionais $P(p_x, p_y)$ e $Q(q_x, q_y)$, a distância é computada como: $\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$.

Definição 4.3.5. ***Directly density-reachable*** Um ponto P é alcançável por densidade diretamente (*directly density-reachable*) de um ponto Q, quando $P \in NEps(Q)$ e $NEps(Q) \geq MinPts$.

Na Figura 4, o ponto A representa um exemplo de *core point*, para o determinado raio *Eps*, $MinPts \geq 7$. O ponto B é um *border point*, pois está dentro do *core point* A e não possui *MinPts* suficientes dentro do *Eps*. O ponto C é um *noise point*, pois não está dentro de nenhuma outra vizinhança e não possui *MinPts* suficientes. O ponto B é alcançável por densidade diretamente do ponto A, mas A não é alcançável através do ponto B, pois B não é *core point*.

Figura 4 – Exemplo de *core point, border point* e *noise point*



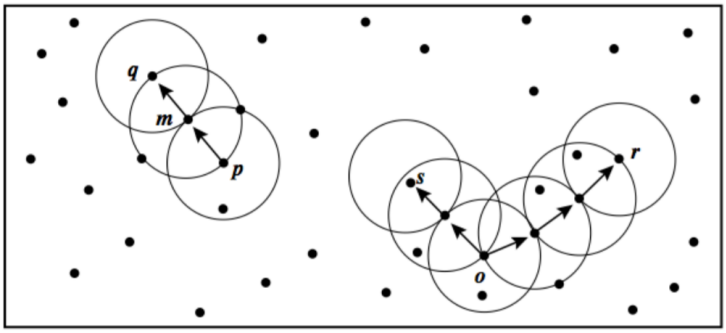
Fonte: Tan, Steinbach e Kumar (2005)

Definição 4.3.6. ***Density-reachable*** Um ponto P é alcançável por densidade (*density-reachable*) de um ponto Q, com respeito à *Eps* e *MinPts* em um conjunto D, se existe uma cadeia de objetos $P_1, ..., P_n$, tais que P_1 = Q e P_n = P e P_{i+1} é alcançável por densidade diretamente de P_i com respeito a *Eps* e *MinPts* para $1 \leq i \leq n$, P_i em D.

Definição 4.3.7. ***Density-Connected*** Um ponto P é conectado por densidade (*density-connected*) à um ponto Q, com respeito à *Eps* e *MinPts* em um conjunto D, se existe um ponto $O \in D$ que tanto P quanto Q são alcançáveis por densidade a partir de O, considerando *MinPts* e *Eps*.

Na Figura 5, para um *MinPts* = 3, o ponto q é alcançável por densidade a partir do ponto p, pois q é alcançável por densidade diretamente através de m, e m é alcançável por densidade diretamente através do ponto p. Os pontos r e s são conectados por densidade através de o, pois existe um ponto o, onde o é alcançável por densidade através de o, e o é alcançável por densidade através de o.

Figura 5 – Exemplo de *density-reachable* e *density-connected*



Fonte: Ester et al. (1996)

Definição 4.3.8. ***Cluster* DBSCAN** Seja D uma base de dados de pontos. Um *cluster* C com respeito à *Eps* e *MinPts* é um subconjunto não vazio de D satisfazendo as seguintes condições:

1. $\forall$ P, Q: se $P \in C$ e Q é alcançável por densidade a partir de P com respeito à *Eps* e *MinPts*, então $Q \in C$
2. $\forall$ P, $Q \in C$: P é conectado por densidade a Q com respeito à *Eps* e *MinPts*.

Um *cluster* DBSCAN é o conjunto de pontos conectados por densidade que é maximal com respeito ao alcance por densidade (TRAN; DRAB; DASZYKOWSKI, 2013).

Como o DBSCAN usa uma definição baseada em densidade de um *cluster*, ele é relativamente resistente a ruído e pode manipular *clusters* de formatos e tamanhos arbitrários. Assim, o DBSCAN pode encontrar mais *clusters* que não puderam ser encontrados usando outros métodos. DBSCAN tem problemas quando os *clusters* têm densidades muito variadas. Ele também tem problemas com dados de alta dimensão porque a densidade é mais difícil de definir para esses dados (TAN; STEINBACH; KUMAR, 2005).

O DBSCAN foi adaptado e utilizado por Palma et al. (2008) para realizar a clusterização dos pontos geográficos para detectar os *stops* e *moves* usando o algoritmo CB-SMoT.

4.4 Algoritmos para a detecção de *stops* e *moves*

Nesta Seção, são apresentados dois algoritmos para detecção de *stops* e *moves*, as definições de cada um e a contextualização deles no trabalho. Esses algoritmos são os mais conhecidos no estado da arte, além de serem bem referenciados por outros autores e usarem metodologias simples para a detecção dos *stops*.

4.4.1 Intersection-Based Stops and Moves of Trajectories

O algoritmo *Intersection-Based Stops and Moves of Trajectories* (IB-SMoT) é um algoritmo proposto por Alvares et al. (2007a). Ele considera a intersecção de uma trajetória com os candidatos a *stops* durante um tempo mínimo de duração, ou seja, as posições da trajetória precisam ter um tempo mínimo de permanência nos candidatos para serem consideradas *stops*, enquanto os *moves* são as demais partes dessa trajetória. Ele é mais usado em aplicações de turismo, pois a velocidade não é necessária. Esse algoritmo busca integrar as trajetórias a uma semântica, a fim de identificar os *stops*.

Na Figura 6 (a), o usuário determina que existem 4 candidatos à *stops*, R_{C_1}, R_{C_2}, R_{C_3} *e* R_{C_4}, e uma trajetória do objeto T representada por um conjunto de pontos espaço-temporais $\langle P_0, ..., P_{15} \rangle$. Cada um desses pontos possuem os atributos *(id, x, y, t)* como informações básicas. O IB-SMoT determina para cada ponto, começando por P_0, em qual candidato se encontra. No início da trajetória, T está fora de qualquer candidato, logo é determinado que a trajetória inicia com um *move*. No ponto P_3, T está dentro do candidato R_{C_1}, então é analisado se o objeto fica tempo suficiente dentro do R_{C_1}. É determinado então que o objeto passou tempo suficiente (ΔC) dentro do candidato. Portanto, R_{C_1} é o primeiro *stop*, pois faz a interseção com os pontos $\langle P_3, ..., P_5 \rangle$ e $\langle P_0, ..., P_3 \rangle$ é o primeiro *move*. Quando o objeto T entra no R_{C_2}, é verificado que ele não permanece tempo suficiente, logo, este candidato não é um *stop*. Depois é analisado o P_{13} que está dentro do candidato R_{C_3}. Verifica-se que T passou tempo suficiente dentro do candidato, portanto $\langle P_5, ..., P_{13} \rangle$ é o segundo *move* e $\langle P_{13}, ..., P_{15} \rangle$ é o segundo *stop*. R_{C_4} é um candidato a *stop*, mas ele não possui interseção com qualquer um dos pontos, então não é analisado e, consequentemente, não é um *stop*.

Para o IB-SMoT, existem os seguinte parâmetros: *User Buffer* e *RF Min Time*. O parâmetro *User Buffer* é representado como sendo, o tamanho do raio da zona ao redor dos pontos de interesse. Esse parâmetro é usado para suprir determinadas incertezas espaciais que venham a acontecer, em outras palavras, é criada uma margem ao redor dos pontos de interesse, buscando minimizar erros e falhas. É possível desativar esse parâmetro, e a unidade de medida desse valor é metros. Os candidatos a *stops* são utilizados nos algoritmos, juntamente com um respectivo tempo de duração mínimo para cada um deles. Esse tempo de duração mínimo é o *RF Min Time*. Não é possível desativar esse parâmetro, e a unidade desse valor é representada em segundos.

4.4.2 Clustering-Based Stops and Moves of Trajectories

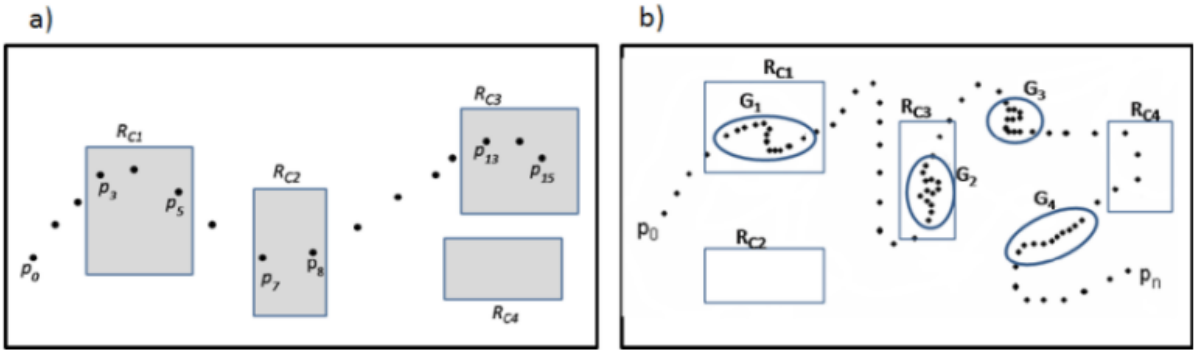
O outro algoritmo que é usado neste trabalho, é o algoritmo *Clustering-Based Stops and Moves of Trajectories* (CB-SMoT). Esse algoritmo foi proposto por Palma (2008) e baseia-se na clusterização, e utiliza as definições usadas pelo DBSCAN, mas com adaptações com base na variação da velocidade dos pontos. O algoritmo é dividido em duas partes principais: na primeira, as partes mais lentas de uma única trajetória são identificadas, chamadas de potenciais *stops* através do método de clusterização do DBSCAN; na segunda parte, o algoritmo identifica onde os potenciais *stops* encontrados (*clusters*). Na primeira parte, estão localizados, considerando os candidatos a *stops*. O algoritmo pega cada potencial *stop* e testa a interseção com os candidatos e se permanece durante um tempo mínimo em cada candidato.

No CB-SMoT, diferentemente do IB-SMoT, se houver um potencial *stop* que não intersecta nenhum dos candidatos, este ainda pode ser considerado um ponto de interesse. O algoritmo define como sendo um *unknown stop*. Um padrão de movimento pode ser gerado para esses *unknown stops*, caso várias trajetórias permaneçam por um período de tempo mínimo no mesmo *unknown stop*. Caso isto venha a acontecer, é necessário investigar e determinar o que seria esse *unknown stop*, pois pode ser um novo ponto de interesse, como, por exemplo: um novo restaurante, um novo bar ou uma rua com algum problema que está dificultando a passagem dos objetos por lá. O CB-SMoT é útil, quando o usuário quer levar em consideração a velocidade dos objetos. Geralmente esse método é usado em aplicações relacionadas ao tráfego urbano, onde é possível identificar regiões com congestionamentos (PALMA, 2008).

Na Figura 6 (b), o usuário determina que existem 4 candidatos à *stops*, R_{C_1}, R_{C_2}, R_{C_3} *e* R_{C_4}, representados por retângulos, e uma trajetória de um objeto T representada por um conjunto de pontos espaço-temporais $\langle P_0, ..., P_n \rangle$. Na primeira parte, o algoritmo encontrou quatro *clusters* e isso significa que encontrou 4 potenciais *stops* representados por elipses: G_1, G_2, G_3 *e* G_4. Na segunda parte, realiza-se uma análise semântica parecida com a realizada pelo IB-SMoT. Observa-se se os *clusters* possuem uma interseção com os candidatos durante um determinado tempo mínimo. Ele analisa e determina que G_1 está intersectando R_{C_1} por um tempo maior que ΔC, que representa o tempo mínimo que ele deve permanecer. Logo, é identificado que R_{C_1} é um *stop*. G_2 está intersectando R_{C_3}, ou seja, G_2 é um *stop*. R_{C_2} e R_{C_4} não possuem nenhum potencial *stop*, logo não são *stops*. G_3 e G_4 que são *clusters* detectados na primeira parte e que não fazem interseção com nenhum outro candidato a *stop* são os definidos anteriormente como *unknown*

stops (ALVARES et al., 2010).

Figura 6 – (a) Exemplo do método IB-SMoT, (b) exemplo do método CB-SMoT



Fonte: Alvares et al. (2010)

O CB-SMoT pretende criar *clusters* de partes de trajetórias em baixa velocidade. Em Palma (2008), são definidos alguns conceitos para a realização do CB-SMoT, baseado-se nos conceitos utilizados do algoritmo do DBSCAN.

Para o CB-SMoT, existem os mesmos parâmetros que no IB-SMoT e novos parâmetros. O parâmetro *User Buff* é utilizado nos pontos de interesse, então sua definição não muda no CB-SMoT. A diferença nos parâmetros que existem no IB-SMoT e no CB-SMoT, é a utilização do *RF Min Time* nos *clusters*, e não nos pontos em si. O parâmetro *MaxAvgSpeed* determina a velocidade máxima que a média de todos os pontos de uma determinada trajetória devem ter para ser considerada um *cluster*. O parâmetro *MinTime* determina o tempo mínimo que uma trajetória tem que ter para ser considerada um *cluster*. O parâmetro *MaxSpeed* determina que a velocidade dos pontos vizinhos de um determinado ponto na análise para a criação do *cluster*, não deve ser superior a essa velocidade. Os parâmetros *MaxAvgSpeed* e *MaxSpeed* representam valores relativos ao valor absoluto de cada trajetória. Os valores representam um percentual da velocidade em km/h para os parâmetros. Por exemplo, se o valor escolhido para *MaxAvgSpeed* foi 0.8, então significa que o valor representa 80% da velocidade média da trajetória. Se a velocidade é de 41,25 km/h, então o valor absoluto para esse parâmetro será de 33 km/h. Neste trabalho, esses valores são representados como km/h.

5 TRABALHOS RELACIONADOS

A seguir, são apresentados os principais trabalhos relacionados.

5.1 Weka-STPM

O *Waikato Environment for Knowledge Analysis* (Weka) é uma coleção de algoritmos avançados de aprendizado de máquina e ferramentas de pré-processamento de dados. Foi projetado para usuários poderem testar rapidamente os métodos de aprendizado de máquina existentes em novos conjuntos de dados de maneira flexível (FRANK et al., 2009).

Alvares et al. (2010) desenvolveram o conhecido Weka-STPM (*Semantic Trajectories Preprocessing Module*), que é uma extensão do Weka. Ele é uma aplicação que permite detectar os *stops* de forma prática por meio do processamento de trajetórias brutas para trajetórias semânticas. Nele, estão implementados os algoritmos CB-SMoT e IB-SMoT. Os algoritmos implementados no Weka-STPM requerem alguns parâmetros para a sua execução.

Todos os parâmetros possuem um valor padrão, mas neste trabalho, foi investigado e determinado os melhores parâmetros para ser utilizados na construção das trajetórias semânticas para os dados do TDrive. Na Figura 7, uma exemplificação dos valores dos parâmetros é mostrada, onde o método selecionado a ser utilizado é o CB-SMoT.

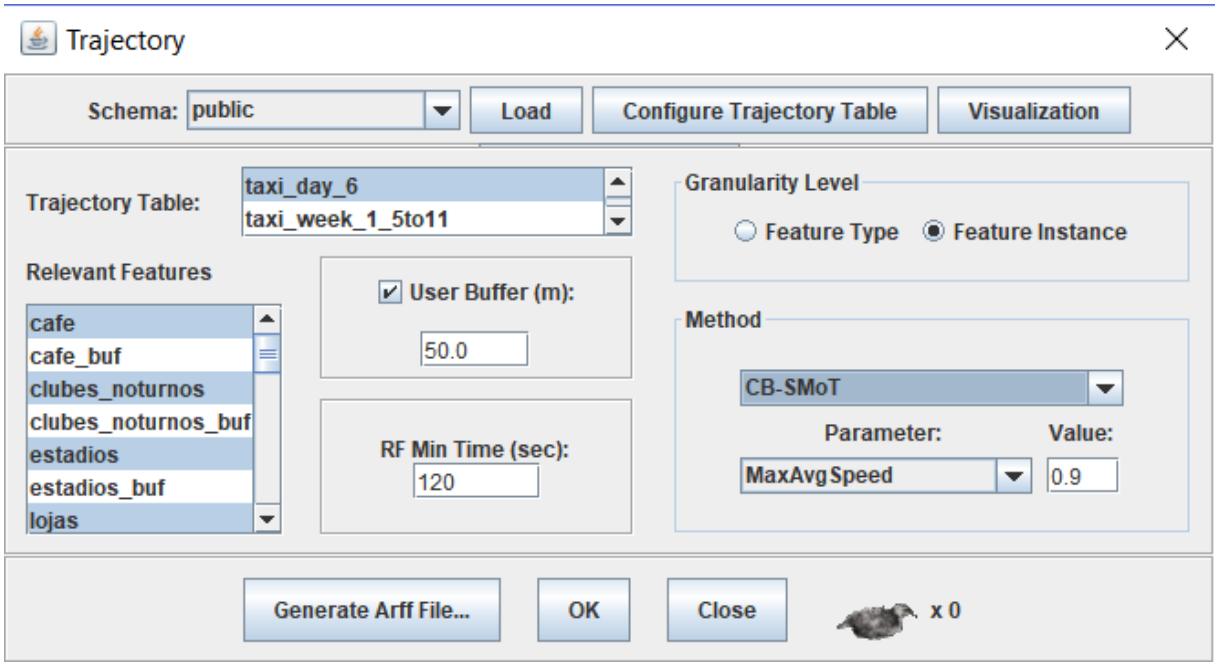
Este trabalho foi construído a partir do trabalho de Alvares et al. (2010), pois foram utilizadas as implementações dos algoritmos de *stops*. Este trabalho utiliza o Weka-STPM, uma implementação de *Map Matching* e uma Interface de Programação de Aplicação (*Application Programming Interface*, API) para encontrar os pontos de interesse da região.

5.2 SeMiTri

O *Semantic Middleware Trajectories* (SeMiTri) é um *framework* que busca enriquecer as trajetórias brutas, através dos dados geográficos fornecidos por terceiros. O enriquecimento das trajetórias semânticas ocorre através de algoritmos de anotação semântica, ou seja, que adicionam informações às trajetórias. Ele foi construído para trabalhar com trajetórias heterogêneas, ou seja, diferentes tipos de objetos em movimento com diversos comportamentos (YAN et al., 2011a).

Yan et al. (2011a) definem que o objetivo do SeMiTri é apoiar o enriquecimento semântico de trajetórias explorando as propriedades geométricas do fluxo, os dados geográficos

Figura 7 – Parâmetros do método CB-SMoT no Weka-STPM



Fonte: Adaptado de Alvares et al. (2010)

e os dados de aplicativos. Esse enriquecimento ocorre a partir de anotações incorporadas nos dados das trajetórias que fornecem conhecimento extra.

Este trabalho propõe diversas funções e processos semelhantes aos existentes no SeMiTri. No SeMiTri é possível criar as trajetórias semânticas, sendo que a principal diferença é a estratégia de identificação dos *stops*.

O SeMiTri foi desenvolvido para ser utilizado com diferentes tipos de objetos em movimento. Neste trabalho busca-se utilizar somente trajetórias de carros. Enquanto o SeMiTri foca mais na parte da criação da semântica. O presente trabalho foca no processamento dos algoritmos para a identificação de *stops* e na adição semântica aos pontos (YAN et al., 2011a).

5.3 SeTraStream

O SeTraStrem é um *framework online* que possibilita a construção de trajetórias semânticas sobre *stream* de dados de movimento. Ele é um dos primeiros trabalhos propostos na literatura que aborda problemas com *stream* em tempo real. A maioria dos métodos existentes para a construção de trajetórias semânticas, utilizam procedimentos *offlines*. Tais métodos não são razoáveis para os aplicativos modernos da vida real, pois os dados de posicionamento dos objetos em movimento são continuamente gerados como *streams* e as operações de consulta

correspondentes, geralmente, exigem a entrega de resultados de maneira *online* e contínua (YAN et al., 2011b).

Yan et al. (2011b) inclui procedimentos de limpeza e compressão dos dados, o que ocorre antes da identificação com precisão dos episódios, os *moves* e *stops*, das trajetórias nos dados de movimentação de *stream* de objetos. Esses procedimentos reduzem a quantidade de trajetórias com erros e os dados das trajetórias que estão crescendo rapidamente, isso ocorre, pois esses dados não devem exceder a capacidade do sistema.

A Tabela 1 mostra uma comparação entre os trabalhos relacionados e este presente trabalho.

Tabela 1 – Comparação entre os trabalhos relacionados e o proposto

	Yan et al. (2011a)	**Yan et al. (2011b)**	**Alvares et al. (2010)**	**Presente trabalho**
Criação de *stops*	Sim	Sim	Sim	Sim
Uso do *map matching*	Sim	Sim	Não	Sim
Algoritmos de enriquecimento semântico	*Framework* SeMiTri	*Framework* SeTraStream	IB-SMoT e CB-SMoT	IB-SMoT e CB-SMoT
Tipo de Dados	Dados históricos	Fluxo de dados	Dados históricos	Dados históricos
API de pontos de interesse	Não identificado	Não identificado	Não identificado	Sim
Disponibilização da aplicação	Não identificado	Não identificado	Sim	Sim

Fonte: Elaborada pelo autor

6 PROCEDIMENTOS METODOLÓGICOS

A seguir, são apresentados os procedimentos metodológicos deste trabalho e a descrição de como são realizados.

6.1 Análise e utilização do *map matching* do *GraphHopper*

Nesse primeiro passo, o *map matching* do *GraphHopper*[1] foi analisado e ajustado com outra pessoa para ser utilizado neste projeto. Esses ajustes foram realizados, buscando reduzir a quantidade de *outliers* e pontos com a posição geográfica errada para a sua utilização. Essa implementação aumenta ou diminui a quantidade de pontos válidos. Não é possível saber ao certo, pois varia conforme a base de dados utilizada. A saída da implementação é usada para os próximos algoritmos adicionarem informações semânticas.

O algoritmo específico que foi ajustado, é o apresentado em Newson e Krumm (2009), uma versão implementada no *GraphHopper*. Esse ajuste foi necessário por causa de problemas na saída do instante de tempo do *GraphHopper*. O objetivo dessa implementação é suprir a falta dos instantes de tempo, baseando-se no *matching* das coordenadas antigas com as processadas, utilizando distância euclidiana entre os pontos para calcular os instantes de tempo restantes.

Os dados tratados são dados de veículos, como carros, mas é possível tratar qualquer tipo de objeto em movimento a partir de procedimentos semelhantes. Através de uma base de dados fornecida, o algoritmo processa essa base de dados em uma nova. Nessa nova base de dados, os pontos estão bem definidos e tem seu posicionamento corrigido para passar exatamente pela rede de ruas, prontos para serem usados pelos próximos algoritmos.

6.2 Estudo do algoritmo IB-SMoT e CB-SMoT

Nesse passo, é realizado um estudo mais detalhado do algoritmo na aplicação de Alvares et al. (2010) para um melhor conhecimento dos métodos da aplicação. Foram realizadas modificações para que a implementação retornasse melhor os *stops* de modo mais preciso, conforme a explicação a seguir. O Weka-STPM apenas detecta os *stops* dos pontos, mas não adiciona semântica aos pontos, então foi necessário alterá-lo para que retornasse os pontos geográficos e uma informação referenciando a qual *stop* ele está associado. Após essas

[1] https://www.graphhopper.com/

modificações, os algoritmos retornam todos os *stops* encontrados, quais são os pontos que criaram esses *stops* e qual o tipo de ponto de interesse é aquele *stop*.

Nessa etapa também é realizada uma busca para identificar os melhores parâmetros para a identificação de *stops*. Os algoritmos IB-SMoT e CB-SMoT possuem diversos parâmetros que influenciam no resultado da identificação dos *stops*. Para que não haja problemas com os parâmetros iniciais, como valores nulos e discrepantes, é necessário testar e definir os melhores valores. Os valores padrões dos parâmetros definidos pelo algoritmo não se adaptam corretamente a todos os tipos de trajetórias. Isso se deve ao fato de, os objetos das trajetórias possuírem velocidades diferentes. Uma possível solução é, calcular e definir um padrão para os tipos de trajetórias e de objetos em movimento.

6.3 Análise da execução dos algoritmos

Nesse passo, os algoritmos são testados para uma determinada base de dados. O motivo disto é realizar testes e verificações das saídas dos algoritmos e da API[2], buscando os interligar de forma manual. Ao término desta etapa, espera-se que os algoritmos estejam funcionando corretamente, e os dados estejam de acordo com o esperado.

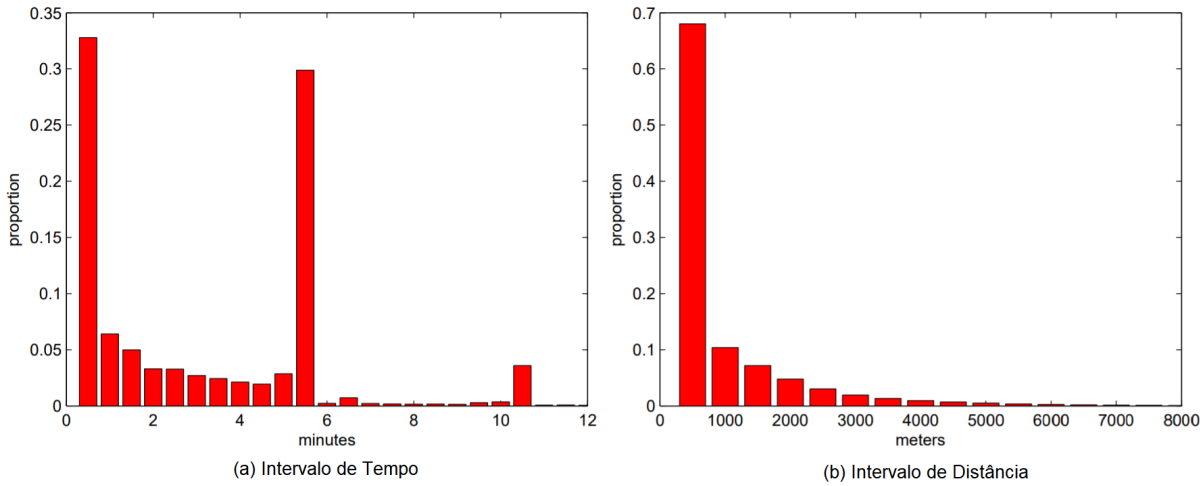
A API foi desenvolvida por outro autor e disponibilizada[3]. A partir de uma entrada dizendo qual a região em que os pontos se encontram, são retornados os pontos de interesse da região. Esses pontos de interesses são utilizados como uma das entradas pelos algoritmos de identificação de *stops*.

Os algoritmos utilizados são uma versão ajustada do *GraphHopper*, para minimizar os erros de precisão, e uma versão ajustada do IB-SMoT e o CB-SMoT para detectar os *stops* nas trajetórias.

A base de dados de trajetórias utilizada neste trabalho foi o T-Drive[4]. Essa base de dados contém a trajetória de GPS de 10.357 táxis, coletados entre o dia 2 de fevereiro e 8 de fevereiro de 2008 na cidade de Pequim, China. Ela possui uma quantidade de aproximadamente 15 milhões de pontos. Na Figura 8 é possível ver a representação dos pontos através de uma distribuição para o intervalo de tempo e de distância entre dois pontos consecutivos. Observa-se que a maioria dos pontos possuem um tempo de distância menor que seis minutos, além disso, a maioria dos pontos também possuem uma distância menor que mil metros (ZHENG, 2011).

[2] https://interest-points.herokuapp.com/
[3] https://github.com/GabrielCzar/api-interest-points
[4] https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/

Figura 8 – Histogramas do intervalo de tempo e da distância entre dois pontos consecutivos



Fonte: Zheng (2011)

6.4 Desenvolvimento de uma aplicação para a execução dos algoritmos de enriquecimento semântico e análise das saídas para a visualização e tomada de decisões

Essa é a etapa onde foi desenvolvida a aplicação. Essa aplicação busca deixar o usuário mais próximo dos processos de enriquecimento semântico. Nessa etapa são definidos os conceitos para o desenvolvimento dela. Para que isto ocorra com perfeição, é necessário que os conceitos e definições dos algoritmos estejam bem compreendidos. Essa aplicação é responsável por executar os algoritmos, adicionando semântica às trajetórias, possibilitando a visualização e análise das saídas para a tomada de decisões.

Essa é uma aplicação WEB que possibilite o usuário executar esses algoritmos para quaisquer base de dados que possua os atributos: *tid, latitude, longitude, time, edge_id, offset, gid* e *the_geom*. O atributo *tid* representa o identificador do objeto em movimento, *time* é o intervalo de tempo do ponto, *edge_id* é o id da aresta do ponto, *offset* pode ser deixado vazio, *gid* é um identificador único para o ponto e *the_geom* é a geometria do ponto.

6.5 Analisar a eficiência e precisão dos resultados dos algoritmos

Após a criação da aplicação, é necessário analisar a qualidade dos resultados, ou seja, identificar o quão bem a trajetória melhorou após o enriquecimento semântico. Nessa etapa, os resultados obtidos foram analisados e ela se baseou em alguns pontos, como por exemplo:

se ainda existem pontos que continuam fora da rede, pontos que a semântica foi adicionada de forma errada, além de problemas com os parâmetros definidos. Essa análise foi realizada manualmente por um especialista por meio da visualização dos resultados. A discussão gerada foi analisada e através dos resultados dessa etapa, modificações, quando necessárias, foram realizadas para consertarem esses problemas.

7 RESULTADOS

A seguir, são apresentados os resultados obtidos em cada etapa dos procedimentos metodológicos deste trabalho e a descrição de como foram realizados.

Os dados das trajetórias do T-Drive foram utilizados como base para este trabalho. A quantidade de pontos foi reduzida para facilitar no processamento dos dados. De 10.357 taxistas foram apenas utilizados os dados de 50 taxistas referente ao dia 3 de fevereiro, resultando em um valor de aproximadamente 75 mil pontos. Na Figura 9, é possível visualizar através de um *heatmap* (mapa de calor), os pontos do dia 3 de fevereiro distribuídos pela cidade de Pequim. Esses pontos foram armazenados em uma tabela no PostgreSQL[1]. Essa tabela apresenta como atributos *tid, date-time, latitude, longitude* e um identificador único para o ponto. O atributo *tid* representa o identificador do táxi, *date-time* representa o tempo que o ponto foi coletado, enquanto *latitude* e *longitude* representam a posição geográfica do ponto. Após a inserção dos dados no PostgreSQL, os algoritmos foram executados em passos, pois a aplicação criada só executa o enriquecimento semântico dos pontos.

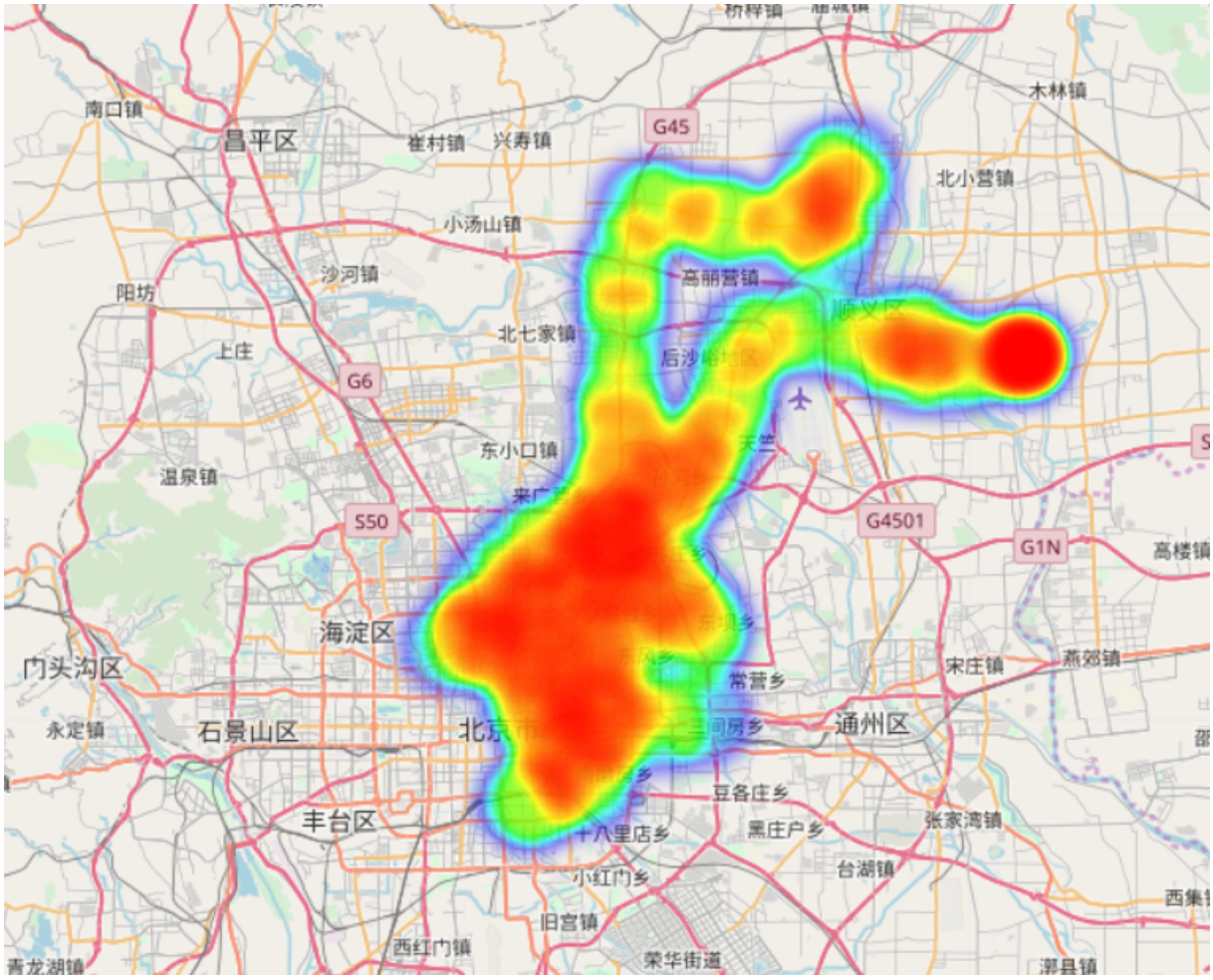
7.1 *Framework* para o enriquecimento semântico

Foi criado um *framework* que inclui os algoritmos e API utilizados neste trabalho, com o objetivo de tornar mais eficientes e compreensíveis as etapas do processo de enriquecimento semântico. Esse *framework* mostra o passo a passo a partir da coleta dos dados até a visualização dos resultados finais, mostrando como os algoritmos estão funcionando em conjunto.

Na Figura 10, é apresentado o *pipeline* do *framework* para o enriquecimento semântico. Na etapa (1), o usuário determina e escolhe uma região para realizar a coleta de dados, além disso, é importante definir o meio de transporte. Em (2), os dados são coletados através de aplicações que capturam as coordenadas geográficas e o intervalo de tempo. Esses dados brutos são armazenados no banco de dados. Antes de realizar o enriquecimento semântico, é necessário resolver os problemas da inconsistência dos dados do GPS usando uma implementação para resolver o *map matching*. Em (3), os dados brutos são utilizados como entrada para a implementação. Eles são processados e a saída é armazenada no banco de dados durante a etapa (4).

[1] https://www.postgresql.org/

Figura 9 – *Heatmap* dos dados dos taxistas do dia 3 de fevereiro

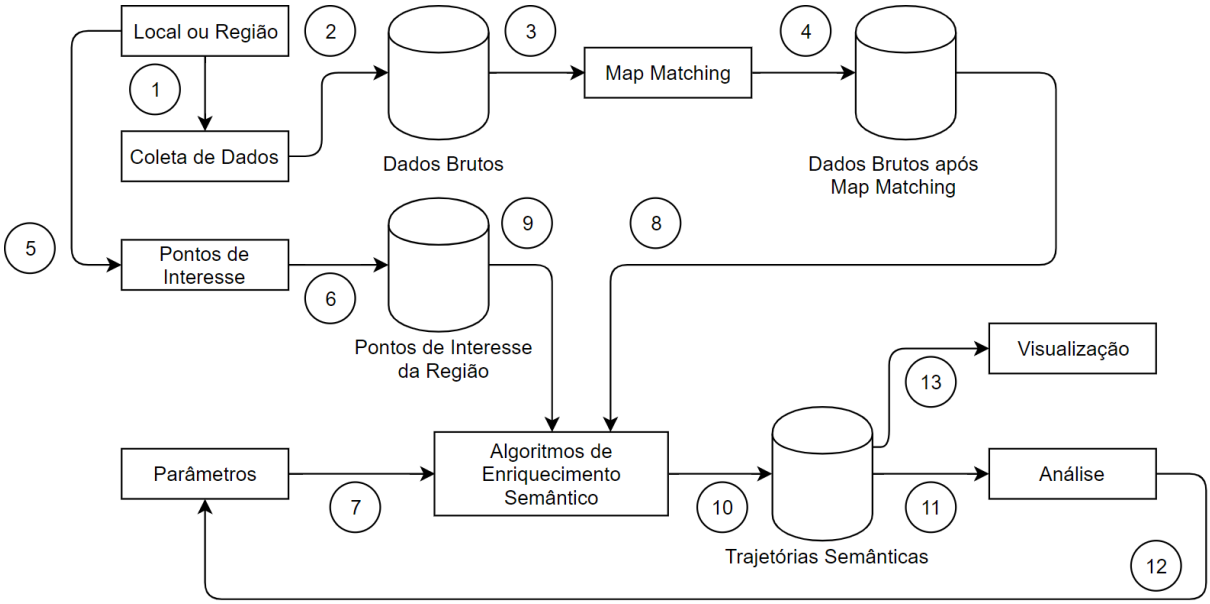


Fonte: Elaborado pelo autor

Após o *map matching*, é necessária a coleta dos pontos de interesses. Para coletar esses pontos, é utilizado, como entrada, o local escolhido (5), a saída é o armazenamento dos pontos de interesse no banco de dados (6). Finalizada essa coleta, o próximo passo é a escolha dos parâmetros para a execução dos algoritmos de detecção de *stops*. O usuário pode escolher os parâmetros através do valor padrão, mas existe a possibilidade dele mesmo definir os seus próprios parâmetros através de experimentos com os dados. No próximo capítulo, o experimento realizado para determinar os melhores parâmetros para uma determinada base de dados é apresentado. Esses parâmetros (7), os dados após o *map matching* (8) e os pontos de interesses (9) são as entradas para os algoritmos de enriquecimento semântico. Após o processamento desses algoritmos, as saídas são armazenadas no banco de dados (10). Logo após, é necessário realizar uma análise (11) para determinar se os resultados foram satisfatórios. Caso não tenham sido ou não tenham gerado resultados, é necessário modificar os parâmetros (12) e

executar o processo novamente a partir da etapa (7). Caso o usuário determine que os resultados foram satisfatórios, os dados podem ser visualizados (13).

Figura 10 – Processo de Enriquecimento Semântico



Fonte: Elaborado pelo autor

7.2 Análise e Utilização do *map matching* do *GraphHopper*

Durante essa etapa, a implementação do algoritmo *map matching* do *GraphHopper* foi analisado e ajustado. Ele é utilizado para alinhar os pontos a rede de ruas e gerar uma pseudo rota entre esses pontos, mas foi necessário realizar um ajuste, pois a versão que era utilizada não retornava os instantes de tempo. O objetivo desse ajuste era suprir a falta dos intervalo de tempo nas coordenadas e padronizar a saída para ser utilizada como entrada para o enriquecimento semântico e outras aplicações que desejem utilizar os resultados dele. Essa modificação foi realizada com outra pessoa e possui código aberto[2].

No algoritmo ajustado é realizado um pré-processamento, nele são eliminados possíveis erros iniciais ou pontos com coordenadas inconsistentes. Essa eliminação é baseada nos pontos com velocidade superior à 150km/h e pontos que estejam fora da área demarcada pelo *OpenStreetMap*[3] (OSM) da cidade.

As bibliotecas e ferramentas usadas para a modificação do *GraphHopper* estão

[2] https://github.com/GabrielCzar/MapMatching
[3] Projeto de mapeamento colaborativo para criar um mapa livre e editável do mundo

definidas no Wik do GitHub do projeto. Na Wiki, também é possível visualizar o passo a passo para a utilização da modificação realizada.

Durante a análise, a implementação do *map matching* foi executada. Todos os pontos são usados nessa etapa como forma de entrada. A saída é uma nova tabela chamada de *matched-tracks* contendo os atributos *tid, latitude, longitude, date-time, edge-id, geom* e o identificador único do ponto. O atributo *tid* representa o identificador do taxi. O atributo *edge-id* representa o número da aresta em que o ponto se encontra na rede de ruas e *geom* é um atributo do tipo geométrico que representa objetos espaciais de duas dimensões. A nova tabela possui aproximadamente 101 mil pontos, pois no terceiro passo do *map matching*, o algoritmo pode criar pontos intermediários para fazer uma melhor conexão dos pontos da trajetória, como, por exemplo, em curvas, mas também pode acontecer de pontos serem eliminados, principalmente no pré-processamento.

Os dados de saída do *map matching* foram visualizados através do QGIS[4], aplicação de sistema de informação geográfica que permite a visualização, edição e análise de dados georreferenciados. Na Figura 11, é exibido um trecho da cidade de Pequim, onde os pontos em azul representam os pontos antes de realizar o algoritmo, os pontos em verde representam os pontos após a execução do algoritmo, as linhas representam as ruas. Alguns dos pontos em azul estão fora das ruas, mas após a aplicação do algoritmo, os pontos são colocados dentro delas, é possível visualizar a criação de mais alguns pontos verdes, pois os pontos originais estavam muito afastados e eles estavam em uma rua com curva. É possível, de maneira visual, observar que houve criações de novos pontos.

Figura 11 – Visualização de um trecho antes (pontos azuis) e depois (pontos verdes) da execução do *map matching*

Fonte: Elaborado pelo autor

[4] https://qgis.org/en/site/

7.3 Estudo do algoritmo IB-SMoT e CB-SMoT

Nesse passo, foram analisadas as implementações dos algoritmos IB-SMoT e CB-SMoT no Weka-STPM. Algumas correções foram realizadas para melhorar a estrutura do código, além de alguns ajustes mínimos para retornar a saída corretamente. O Weka-STPM apenas detecta os *stops*, então foi necessário modificá-lo para adicionar a semântica nos pontos. A nova versão com essa modificações e ajustes foi publicada em um repositório do GitHub[5].

Após a análise do código do Weka-STPM, foi necessário analisar os parâmetros dos algoritmos. Essa análise foi realizada com o objetivo de determinar os melhores parâmetros para a identificação de *stops* para carros. Os algoritmos do Weka-STPM possuem alguns parâmetros que influenciam na quantidade e qualidade dos *stops*. Então esses *stops* foram analisados graficamente e por consultas espaciais. Vale ressaltar que outras variáveis ainda podem influenciar nos parâmetros, como a frequência de obtenção dos pontos de GPS. Caso seja utilizado para outros tipos de objetos de movimento, pode-se utilizar dos mesmos procedimentos para descobrir novos parâmetros para esses objetos.

A variação dos parâmetros ocorreu para os parâmetros *User Buff* e *RF Min Time*. Foram determinados, de acordo com testes prévios, cálculos para encontrar a melhor área ao redor do ponto de interesse. Também foi observando os padrões determinados pelo Weka-STPM na sua implementação original. Na Tabela 2, são apresentados os valores variados para o *User Buff* e para o *RF Min Time*. Determinados os valores dos dois parâmetros do IB-SMoT, para cada valor escolhido para o primeiro parâmetro variou-se todos os valores do segundo. Uma quantidade total de 12 testes foram realizados para o algoritmo do IB-SMoT. A análise desses testes também basearam-se nos fatores de tempo de processamento do algoritmo e na quantidade de *stops* encontrados. Os resultados podem ser verificados na Tabela 3.

Tabela 2 – Parâmetros escolhidos para o *User Buff* e o *RF Min Time*

Parâmetro	Valores			
User Buff (m)	100	150	200	
RF Min Time (s)	60	90	120	180

Fonte: Elaborado pelo autor

Na Tabela 3, as colunas representam, respectivamente da esquerda para a direita, o caso de teste, o valor usado em metros para o parâmetro *User Buff*, o valor em segundos para

[5] https://github.com/lucivanbatista/Weka-STPM

o *RF Min Time*, a quantidade em unidades de *stops* encontrados e o tempo de processamento do algoritmo com esses parâmetros. Através desses experimentos, chegou-se em algumas conclusões sobre os parâmetros.

Na análise sobre a quantidade de *stops* encontrados, o *User Buff* é o utilizado para determinar o raio dos pontos de interesse. Quanto maior o valor do *User Buff*, maior é a quantidade de *stops* que são encontrados, ou seja, seu valor é diretamente proporcional a quantidade de *stops* encontrados. O *RF Min Time* é o usado para determinar o tempo de duração mínima do objeto dentro do ponto de interesse para serem considerados *stops*. Quanto maior o valor do *RF Min Time*, mais restrito será para os pontos serem considerados *stops*, logo, ele é inversamente proporcional a quantidade de *stops*.

Com relação ao tempo de processamento dos algoritmos, quanto maior o valor do *User Buff* e do *RF Min Time*, maior é o tempo de processamento para encontrar os *stops*. Ambos influenciam diretamente nisso, mas é notável, através dos testes, que o *User Buff* influência mais que o *RF Min Time*. Por exemplo, no teste (1) e (9) da Tabela 3, o *RF Min Time* não é variado, mas o valor usado para o *User Buff*, respectivamente, foi 100 e 200. Nestes casos, o tempo de processamento dobrou quando o valor do *User Buff* dobrou. No teste (1), (2), (3), (4), o *User Buff* foi mantido, variando o *RF Min Time*, o tempo de processamento não variou muito, sendo a diferença de aproximadamente 25 segundos. Através dessas análises, confirmou-se que o *User Buff* afeta mais que o parâmetro *RF Min Time*.

Após a análise dos testes realizados acima, foi necessário determinar o melhor teste. Usando algumas localizações de pontos de interesse aleatórios e o Google Maps[6], verificou-se em um mapa real a área que está sendo utilizada para os pontos de interesse. Na Figura 12, o ponto azul claro representa um ponto de interesse, enquanto o círculo vermelho, azul escuro e roxo, representam, respectivamente, a área do *stop* criado com o valor de *User Buff* 100, 150 e 200. Esses *stops* foram criados com o valor do *RF Min Time* sendo 60. Confirmou-se que um raio de 100 metros é muito pequeno, a área de alguns pontos de interesse com esse raio não chegam nem perto das ruas, enquanto 200 metros possui uma área extensa. Uma área muito grande pode ocasionar um problema muito comum, esse problema se refere ao caso de haver um ponto de interesse próximo a outro ponto de interesse e suas áreas estarem sobrepostas. Quando uma trajetória passar próximo a eles, não seria possível identificar a qual pertence. Para reduzir esses casos, nesse teste, optou-se pelo valor de 150 metros para o *User Buff*. Essas visualização

[6] https://www.google.com/maps

Tabela 3 – Casos de teste do IB-SMoT

Caso de Teste	User Buff (m)	RF Min Time (s)	Quantidade (u)	Time (ms)
1	100	60	331	106.435
2	100	90	258	125.929
3	100	120	235	113.367
4	100	180	210	132.707
5	150	60	744	187.626
6	150	90	503	179.123
7	150	120	428	185.243
8	150	180	358	186.390
9	200	60	1280	241.239
10	200	90	839	243.008
11	200	120	685	260.679
12	200	180	536	260.972

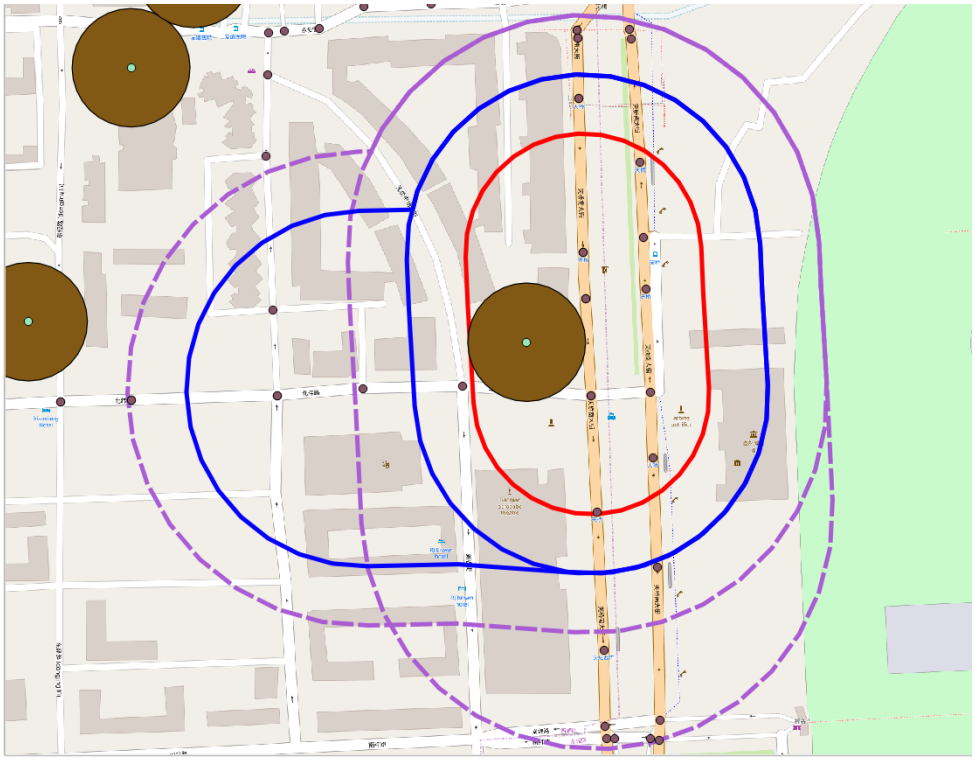
Fonte: Elaborado pelo autor

foram realizadas através do QGIS.

Para o *RF Min Time*, os valores de 180 segundos e de 60 segundos não foram considerados os melhores, pois 180 segundos é muito tempo para um objeto ficar na área do ponto de interesse. Por exemplo, um taxista não fica mais de 3 minutos para embarque e desembarque de passageiros. Enquanto para 60 segundos, qualquer objeto que demorasse 1 minuto na área do ponto de interesse, seria considerado um *stop*, resultando no problema da geração de uma grande quantidade de *stops* que talvez não sejam realmente um *stop*. A escolha foi de 90 segundos (6), um tempo considerável para ser considerado um *stop*. O valor de 120 segundos também poderia ter sido escolhido, sendo que o critério de escolha de 90 segundos, foi porque ele retornou uma quantidade maior de *stops* do que o teste com 120 segundos (7). Determinou-se que o melhor caso, para essa base de dados e para esse tipo de objeto, é o caso de teste (6), onde o *User Buff* é 150 metros e o *RF Min Time* é 90 segundos.

Após os testes e análise do IB-SMoT, foram realizados procedimentos parecidos para o CB-SMoT. O CB-SMoT utiliza, além dos mesmo parâmetros que o IB-SMoT, os parâmetros *Min Time, MaxAvgSpeed* e *MaxSpeed*, eles são os usados na clusterização. Como foram determinados os melhores parâmetros para o IB-SMoT, e a segunda parte do CB-SMoT é realizada o mesmo procedimento que o IB-SMoT, então buscou-se analisar apenas os

Figura 12 – Exemplo de Ponto de Interesse com Área do *User Buff*



Fonte: Elaborado pelo autor

parâmetros da clusterização.

Foram realizados dois experimentos. No primeiro experimento, não foram obtidos bons resultados, pois foram utilizados os valores padrões do Weka-STPM, que foi 50 metros para o *User Buff* e 120 segundos para o *RF Min Time*, não houve variação nos resultados. No segundo experimento, foram utilizados os melhores valores encontrados no IB-SMoT. Estes valores foram fixados e então modificou-se os valores da clusterização.

Os valores escolhidos para os parâmetros *Min Time, MaxAvgSpeed* e *MaxSpeed* estão definidos na Tabela 4. Para cada parâmetro, os valores foram fixados, enquanto os outros foram variados, resultou-se em um total de 18 testes. Na Tabela 5, as colunas representam, respectivamente da esquerda para a direita, o número de teste, o valor usado em segundos para o parâmetro *Min Time*, o valor em km/h para o *MaxAvgSpeed*, o valor em km/h para o *MaxSpeed*, a quantidade em unidades de *stops* total encontrados, a quantidade em unidade de *stops* válidos, ou seja, desconsiderando o *unknown stops*, e o tempo de processamento do algoritmo com esses parâmetros.

Com relação ao tempo de processamento desses testes, não foi verificado uma grande diferença no tempo. A diferença entre o maior e o menor tempo foi de aproximadamente 15

Tabela 4 – Parâmetros escolhidos para o *Min Time, MaxAvgSpeed* e *MaxSpeed*

Parâmetro	Valores		
Min Time (s)	40	60	90
MaxAvgSpeed (km/h)	0.4	0.6	
MaxSpeed (km/h)	0.8	1.0	1.5

Fonte: Elaborado pelo autor

segundos, enquanto nos testes do IB-SMoT, a diferença do tempo foi relativamente maior entre os seus resultados. Pode-se concluir que não houve indícios para esta base de dados, que houve impacto no CB-SMoT de alteração dos parâmetros no tempo de processamento.

O *MaxAvgSpeed* é o principal parâmetro que influencia na quantidade de *stops* encontrados, quanto maior o seu valor, maior é a quantidade encontrada. Baseando-se nisso, foi escolhido o valor 0.6 km/h para ele. O *MinTime* é o tempo mínimo que a trajetória tem que ter para fazer um *clusters*, é inversamente proporcional a quantidade de *stops*, ou seja, diminuir o valor dele, significa ser menos rigoroso e mais trajetórias serem *clusters*. Buscou-se ser menos rigoroso com relação ao tempo, então foi escolhido o valor de 40 segundos. O *MaxSpeed* não influencia muito na quantidade de *stops*, é ele que determina um limite de velocidade para os pontos da trajetória dos *clusters*, como a diferença é pequena, foi escolhido o que possui mais *stops*, então foi escolhido o valor de 1.5 km/h para ele. O valor para o *MaxSpeed* depende bastante do objeto e do intervalo de tempo em que os pontos foram gravados. Para tentar suprir esse problema com a identificação de *stops*, resolveu-se usar o valor de 1.5 km/h. É importante destacar que, o valor do *MaxSpeed* tem que ser maior que o valor do *MaxAvgSpeed*.

Mesmo com a escolha do valor 1.5 km/h para o *MaxSpeed*, uma análise foi realizada para ter a certeza de usar esse valor. Para isso, apenas os testes com o *MinTime* de 40 segundos e o *MaxAvgSpeed* de 0.6 km/h, foram utilizados. Os testes analisados foram o (4), (5) e (6). Para analisar esses testes, utilizou-se do QGIS para uma visualização gráfica e de consultas no banco de dados para uma análise espacial. Com o QGIS, os resultados total de *stops* desses testes foram analisados em conjunto, buscando localizar as diferenças e semelhanças dos resultados. Notou-se que os testes possuem quase os mesmos resultados para o seu total, com exceção do (6) que possui a maioria dos resultados do (4) e (5), e mais alguns. Nesse teste, o que obteve mais resultados foi o caso (6). Analisar a qualidade dos resultados é um problema, pois não há uma maneira fácil de como determinar que aquele resultado é um *stop* válido. Após a análise visual, foi necessário realizar uma análise utilizando de consultas no banco de dados PostgreSQL. Essas

Tabela 5 – Casos de teste do CB-SMoT

Caso de Teste	MinTime (s)	MaxAvgSpeed	MaxSpeed	Quantidade (u)	Qtd Válidas (u)	Time (ms)
1	40	0.4	0.8	690	304	90.095
2	40	0.4	1.0	690	309	93.104
3	40	0.4	1.5	695	306	94.113
4	40	0.6	0.8	970	481	96.873
5	40	0.6	1.0	974	493	101.789
6	40	0.6	1.5	983	490	96.170
7	60	0.4	0.8	676	298	92.485
8	60	0.4	1.0	677	303	98.026
9	60	0.4	1.5	683	300	92.240
10	60	0.6	0.8	950	474	98.481
11	60	0.6	1.0	955	486	98.753
12	60	0.6	1.5	963	483	105.526
13	90	0.4	0.8	663	295	89.535
14	90	0.4	1.0	666	300	100.500
15	90	0.4	1.5	670	297	104.413
16	90	0.6	0.8	932	470	95.134
17	90	0.6	1.0	938	482	98.031
18	90	0.6	1.5	942	479	95.309

Fonte: Elaborado pelo autor

consultas mostraram que a maioria dos resultados são os mesmos, abaixo é detalhado melhor os resultados para a escolha do caso (6) com o valor 1.5 km/h para o *MaxSpeed*.

Após a escolha dos melhores parâmetros para o IB-SMoT e para alguns do CB-SMoT, foi analisado os resultados dos seus melhores casos, o (6) para o IB-SMoT e o (4), (5) e (6) para o CB-SMoT. O objetivo dessa análise, é identificar a quantidade de resultados que ambos possuem em comum, as diferenças, qual possui mais e qual possui menos *stops*. Para isso, utilizou-se de consultas no banco de dados através de operações de associação para obter a interseção dos resultados dos testes e determinar a quantidade que cada um possui de diferente do outro.

O caso (6) do IB-SMoT possui um total de 503 *stops*, sendo que 440 são resultados distintos, essa distinção foi realizada com o objetivo de remover a geometria de lugares repetidos. Para os casos do CB-SMoT, foi utilizado todos os resultados, tanto *unknown stop*, como os válidos. Para cada caso do CB-SMoT, foi comparado os resultados com o caso (6) do IB-SMoT.

Na Tabela 6, tem-se da esquerda para a direita, o número que do caso do CB-SMoT que está sendo comparado ao caso (6) do IB-SMoT, a quantidade total dos resultados, a quantidade total de distintos destes resultados, a quantidade de resultados válidos, ou seja, não *unknown stops*, os resultados que a geometria estão na interseção dos ambos os casos, os

resultados que aparecem apenas em IB-SMoT e os que aparecem apenas no do CB-SMoT.

O caso (6) do CB-SMoT apresenta uma quantidade maior que os outros casos, além de possui uma quantidade maior de valores distintos. Esse caso também possui uma quantidade maior de valores em comum ao caso (6) do IB-SMoT, tornando os resultados mais próximos, mas não os mesmos. Através desta análise, foi possível mostrar que o caso (6) do CB-SMoT, entre os casos apresentados aqui, é o mais relevante e que traz resultados mais próximos com o caso (6) do IB-SMoT, fortalecendo a confirmação dos resultados serem bons *stops*. Foi mostrado que o valor 1.5 km/h, nesses testes, é o melhor valor para o *MaxSpeed*.

Tabela 6 – Comparação do caso (6) do IB-SMoT e dos casos (4), (5) e (6) do CB-SMoT

Caso do CB-SMoT	Total	Total Distintos	Válidos	Interseção com IB-SMoT	Exclusivos do IB-SMoT	Exclusivos deste caso
4	970	794	481	336	104	458
5	974	797	493	346	91	451
6	983	808	490	351	89	457

Fonte: Elaborado pelo autor

Em Palma (2008), é realizado um experimento semelhante ao apresentado anteriormente, mas voltado apenas para o CB-SMoT. A maior semelhança foi na utilização de uma representação visual, além da variação dos valores dos parâmetros exclusivos do CB-SMoT. Nesse artigo, as análises apresentadas são bastante semelhantes às encontradas neste trabalho. É importante ressaltar que os trabalhos comparados não utilizam a mesma base de dados, mas utiliza o mesmo tipo de objeto de movimento. Na Tabela 7, são apresentado os melhores valores deste presente trabalho e dos encontrados em Palma (2008). A maior diferença na comparação dos trabalhos, é com relação ao valor encontrado como melhor para o parâmetro *MaxSpeed*. Os valores mais próximos encontrados na comparação, foi para o parâmetro *MaxAvgSpeed*. O valor do *Min Time* em Palma (2008) foi um dos valores usados para testar esse parâmetro neste trabalho.

Tabela 7 – Comparação dos melhores valores dos parâmetros do *CB-SMoT*

Parâmetro	Presente Trabalho	Palma (2008)
Min Time (s)	40	60
MaxAvgSpeed (km/h)	0.6	0.5
MaxSpeed (km/h)	1.5	0.7

Fonte: Elaborado pelo autor

7.4 Análise da execução dos algoritmos

Nesta etapa, cada um dos algoritmos foi executado para verificar as saídas, buscando interligar seus resultados de forma manual. Primeiro, o *map matching* foi executado com os dados do T-Drive. Todos os pontos são usados nessa etapa como forma de entrada. A saída gerada é uma nova tabela chamada de *matched-tracks* contendo os atributos: *tid, latitude, longitude, date-time, edge-id, geom* e o identificador único do ponto. O atributo *tid* representa o identificador do táxi. O atributo *edge-id* representa o número da aresta em que o ponto se encontra na rede de ruas e *geom* é um atributo do tipo geométrico que representa objetos espaciais de duas dimensões. A tabela de entrada possui um total de 75 mil pontos e a nova tabela possui aproximadamente 101 mil pontos, pois no terceiro passo do *map matching*, pode acontecer do algoritmo descartar pontos por não fazerem conexões ou ser necessário criar pontos intermediários para fazer uma melhor conexão dos pontos, como, por exemplo, em curvas.

Após a execução do *map matching*, foi realizado o procedimento para determinar os pontos de interesse da região da base de dados. A API se baseia no *Overpass turbo*[7], uma ferramenta web para filtrar dados do OSM. A API, aqui utilizada, filtra os pontos de interesses do *Overpass turbo* após a seleção da cidade da base de dados. A Figura 13 mostra a API utilizada.

Figura 13 – API utilizada para a identificação dos pontos de interesse

Fonte: Elaborado pelo autor

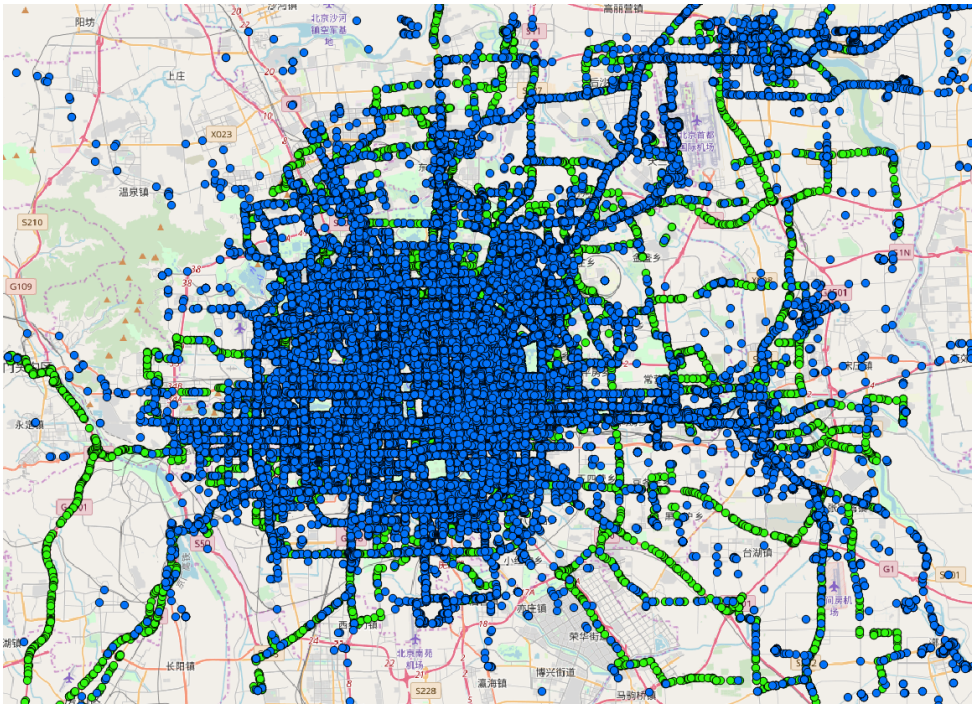
A saída da API é um *JavaScript Object Notation*[8] (JSON) com as informações necessárias sobre os pontos de interesses da região selecionada. Esse JSON é inserido em uma

[7] http://overpass-turbo.eu/
[8] http://www.json.org/

tabela no banco de dados representando os pontos de interesse com os atributos *amenity, name, latitude, longitude, geom* e o identificador único do ponto. *Amenity* representando o tipo do ponto de interesse, como, hotel e restaurante, enquanto *name* representa o nome do ponto de interesse. Após esse processo, um total de aproximadamente 3 mil pontos de interesses foram identificados.

Na Figura 14, são mostrados os pontos dos taxistas na cidade de Pequim e os pontos de interesse da cidade. A cor azul representa os pontos antes do *map matching*. Os da cor verde representam os pontos após o *map matching*. Através da análise baseada na visualização, é possível determinar que os pontos estão dentro das ruas de forma correta.

Figura 14 – Visualização dos pontos antes (pontos azuis) e depois (pontos verdes) do *map matching*

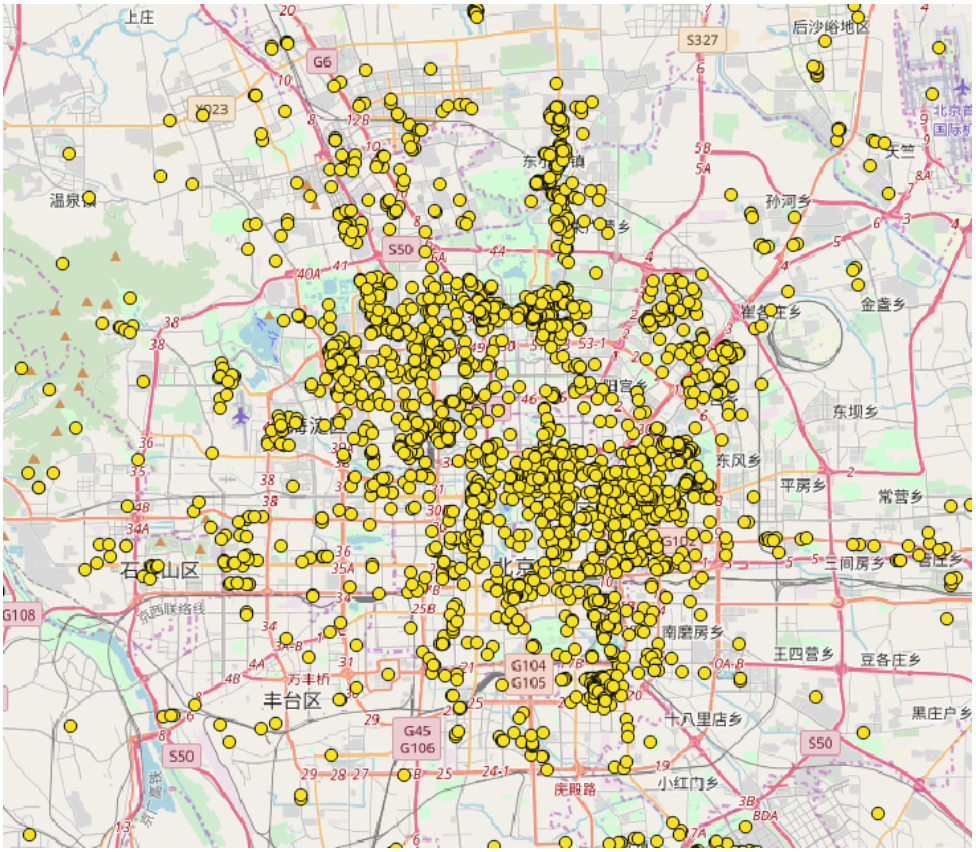


Fonte: Elaborado pelo autor

Na Figura 15, são mostrados da cor amarela, os pontos de interesse identificados da cidade de Pequim. Não é possível saber corretamente se todos os pontos de interesses foram capturados, pois nem todos os pontos de interesses da região estão disponíveis na API, além de existem diversos pontos que não possuem nome, ou alguma das coordenadas, esses são removidos.

O último passo é a detecção dos *stops* e o enriquecimento semântico nos pontos após

Figura 15 – Visualização dos pontos de interesse

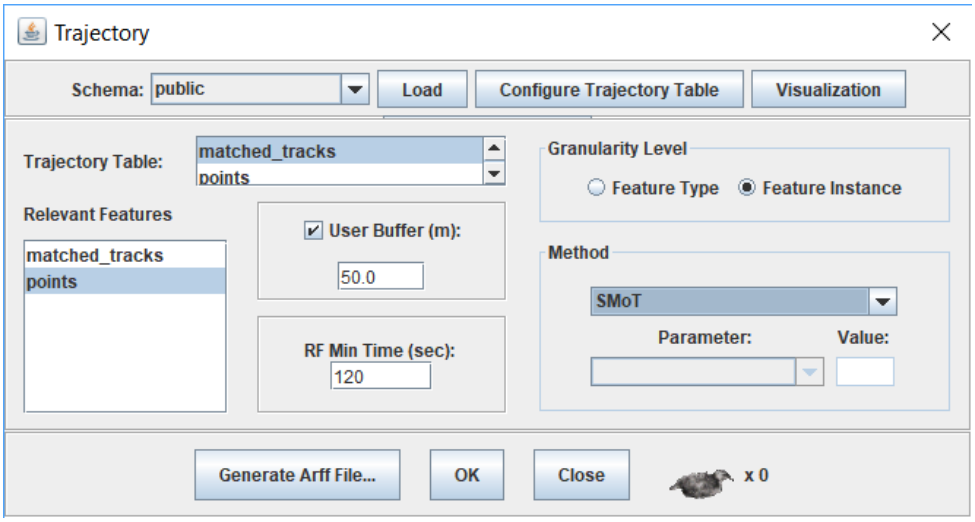


Fonte: Elaborado pelo autor

o processamento do *map matching*. O Weka-STPM utiliza esses dados e os pontos de interesses gerados pela API para a execução dos algoritmos CB-SMoT e IB-SMoT. Vale ressaltar que, os valores dos parâmetros utilizados são os melhores valores encontrados anteriormente para cada algoritmo. A Figura 16 representa a aplicação do Weka-STPM configurado para a execução do método IB-SMoT.

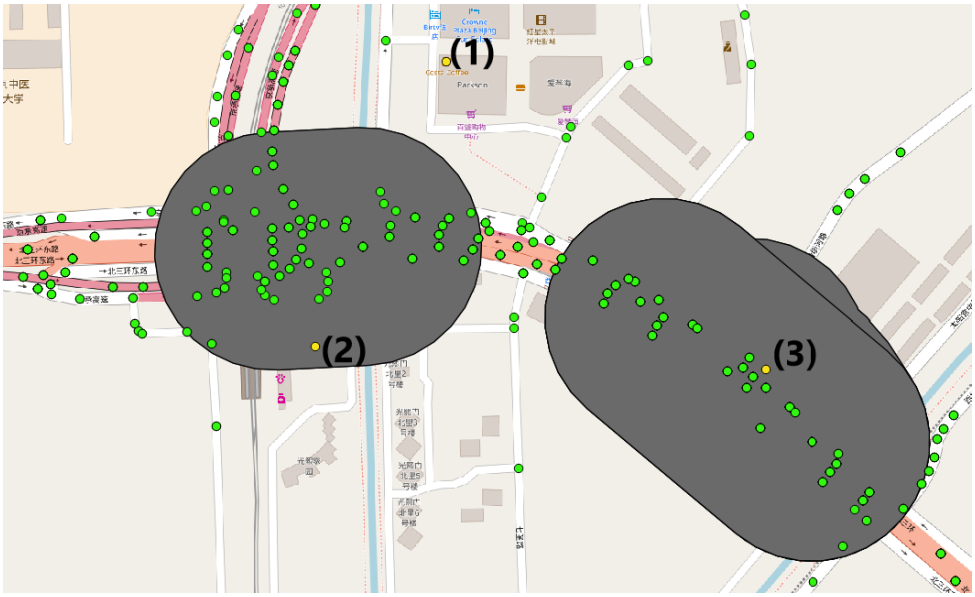
Após a execução dos dois métodos, duas novas tabelas foram geradas, uma para os *stops* do IB-SMoT e outra para os do CB-SMoT. No método IB-SMoT, foram gerados 503 *stops*, enquanto no CB-SMoT, foram gerados 983, sendo 490 válidos e o restante *unknown stop*. Na Figura 17, é mostrado um trecho dos resultados, nele é possível identificar 3 pontos de interesse e 3 *stops*. As geometrias em cinza representam os *stops* encontrados pelo CB-SMoT. Tem-se que o ponto de interesse (3) possui 2 *stops*, ou seja, existem duas trajetórias de dois objetos em movimento que foram identificados. Enquanto isso, o ponto de interesse (1) não possui nenhum *stop*. Nesse trecho não é possível visualizar a geometria, mas o IB-SMoT gerou os mesmos *stops* que o CB-SMoT.

Figura 16 – Aplicação do Weka-STPM para o método IB-SMoT



Fonte: Adaptado de Alvares et al. (2010)

Figura 17 – Visualização de um trecho dos *stops* do CB-SMoT



Fonte: Elaborado pelo autor

7.5 Demonstração do *Framework* para o enriquecimento semântico

Para demonstrar como é realizado o fluxo de trabalho para o enriquecimento semântico, um resumo dos passos é realizado nesta Seção.

Neste trabalho foram utilizados dados de taxistas da cidade de Pequim (etapa 1). Esses dados possuem um total de mais de 15 milhões de pontos (etapa 2) de 10.357 taxistas durante os dias 2 e 8 de fevereiro de 2008, onde cada ponto possui um id para o taxista, o instante

de tempo, a longitude e a latitude, esses pontos foram armazenados em um banco de dados. Foi escolhido apenas um dia, o motivo disso foi devido a base completa ser muito grande e requerer um alto grau de processamento. O dia de domingo foi escolhido, pois é um dia em que a maioria das pessoas estão de folga do trabalho ou da escola e o utilizam para visitar pontos de interesses da cidade. Baseado nesse caso, apenas o dia 3 foi utilizado. (ZHENG, 2011).

A implementação do algoritmo para resolver o problema da inconsistência dos pontos usada neste trabalho, é uma modificação do *GraphHopper*. Os dados dos taxistas são a entrada para essa implementação (etapa 3), a saída são os dados após o processamento (etapa 4). Para a coleta dos pontos de interesse de Pequim, foi utilizado uma API[9] que usa como entrada a região ou local escolhido inicialmente (etapa 5) e retorna os pontos encontrados, inserindo-os no banco de dados (etapa 6).

Para o enriquecimento semântico, foi utilizado o Weka-STPM que implementa o IB-SMoT e o CB-SMoT. Como entrada, utilizam os dados após o *map matching* (etapa 8), os pontos de interesse (etapa 9) e os parâmetros (etapa 7). Para ambos os métodos, os parâmetros são o tempo de permanência mínima (*RF Min Time*) do ponto dentro do ponto de interesse, o raio da área (User Buff) do ponto de interesse, esse raio serve como uma margem de erro. Para o CB-SMoT, é necessário outros parâmetros para o DBSCAN, que são o *MinTime*, *MaxAvgSpeed* e o *MaxSpeed*. Após diversos testes e experimentos realizados, chegou-se na conclusão de que os melhores parâmetros para a nossa base de dados são os valores 150m e 90s para o *User Buff* e o *RF Min Time*, respectivamente, enquanto para o *MinTime*, *MaxAvgSpeed* e o *MaxSpeed*, os melhores valores são 40 s, 0.6 km/h e 1.5 km/h. Para a validação desses parâmetros, diversos critérios foram utilizados, como quantidade de *stops* detectados, tempo de processamento dos algoritmos e *unknown stops* para o CB-SMoT. A saída deste processo são os *stops* (etapa 10), eles são armazenados no banco de dados.

A análise (etapa 11) dos *stops* foi realizada diversas vezes para escolher os melhores parâmetros (etapa 12), esse processo se repetiu diversas vezes até obtermos os melhores resultados através das análises dos parâmetros. No final, é possível visualizar (etapa 13) os resultados em uma ferramenta, como o QGIS.

[9] https://interest-points.herokuapp.com/

7.6 Desenvolvimento de uma aplicação para a execução dos algoritmos de enriquecimento semântico e análise das saídas para a visualização e tomada de decisões

Uma aplicação WEB foi criada para melhor se adaptar ao Weka-STPM. Na Figura 18, vemos o visual da interface com os parâmetros e uma descrição breve e simples para o usuário, caso hajas dúvidas, entender. Nessa tela, é possível enriquecer semanticamente a base de dados escolhida. Na Figura 19, é uma tela que é possível o usuário visualizar em uma tabela os resultados gerados após o enriquecimento semântico. A aplicação consegue abranger as etapas (7) a (13) do framework.

Figura 18 – Interface da aplicação WEB para o enriquecimento semântico

Fonte: Elaborado pelo autor

Figura 19 – Interface da aplicação WEB para a visualização dos *stops*

Enriquecimento Semântico

Consiga enriquecer semanticamente sua base de dados

ib_stops_teste ▾ 20 ▾ Filtrar

Pontos e seus Stops Identificados da tabela ib_stops_teste

Stop ID	Point ID	ID do Objeto	Lat	Lon	Time	Edge	POI ID	Start Time Stop	End Time Stop	Amenity
1	64169	29	39.92260765111315	116.29407144250135	2008-02-04 00:16:18.915	5610	483	2008-02-04 00:09:00.0	2008-02-04 00:16:18.915	hospital
1	64168	29	39.92261342531324	116.29371325583115	2008-02-04 00:09:00.0	5610	483	2008-02-04 00:09:00.0	2008-02-04 00:16:18.915	hospital
2	65321	29	39.94803573820075	116.41949917568637	2008-02-04 12:00:03.815	47098	2625	2008-02-04 11:53:44.624	2008-02-04 12:00:03.815	bank
2	65320	29	39.94802959147162	116.41930993093497	2008-02-04 11:56:12.401	51501	2625	2008-02-04 11:53:44.624	2008-02-04 12:00:03.815	bank
2	65319	29	39.94802642497479	116.41918904526209	2008-02-04 11:53:44.624	41187	2625	2008-02-04 11:53:44.624	2008-02-04 12:00:03.815	bank
3	65331	29	39.948079696627254	116.42462629283874	2008-02-04 13:07:53.64	41183	2141	2008-02-04 12:42:47.39	2008-02-04 13:07:53.64	school
3	65330	29	39.948103911014734	116.42404663765535	2008-02-04 13:07:44.53	41183	2141	2008-02-04 12:42:47.39	2008-02-04 13:07:53.64	school
3	65329	29	39.948068893285146	116.42329003117882	2008-02-04 13:07:32.635	41183	2141	2008-02-04 12:42:47.39	2008-02-04 13:07:53.64	school

Fonte: Elaborado pelo autor

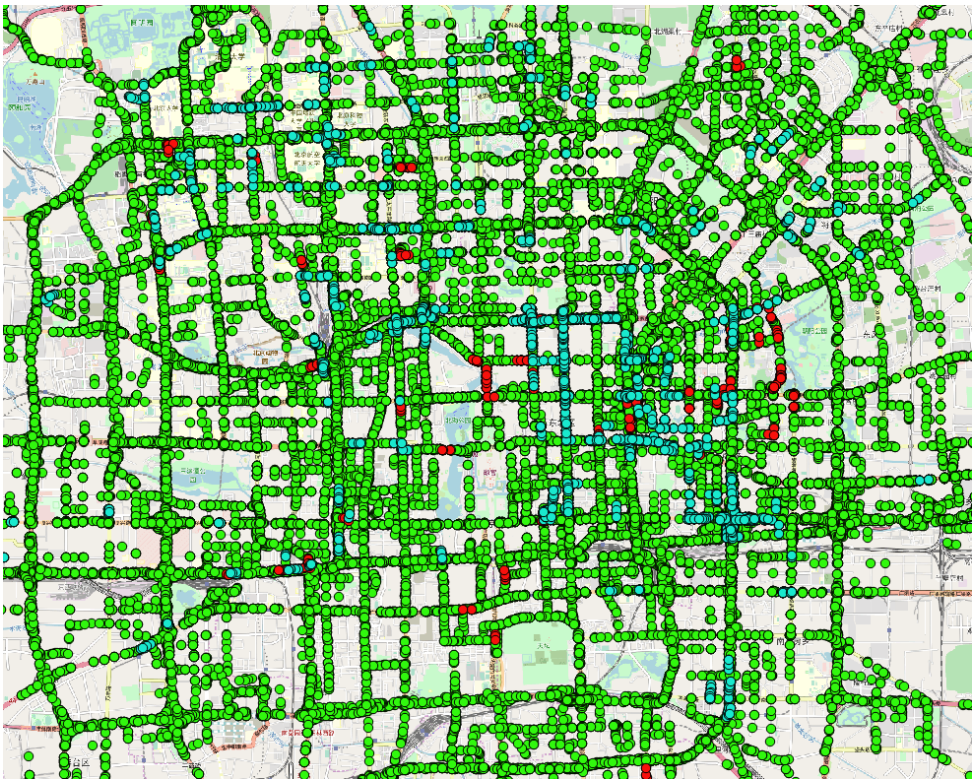
7.7 Analisar a eficiência e precisão dos resultados dos algoritmos

A partir da visualização dos dados após o enriquecimento semântico, é possível determinar que os pontos estão dentro das rede de ruas. É possível determinar quais pontos são *stops* e quais pontos são *moves*, além de ter a área que é considerada um *stop* e a qual ponto de interesse esses pontos estão. Também confirma-se que os parâmetros foram bem definidos e conseguiram gerar bem os resultados dos *stops*. Através desses pontos, pode-se determinar que as trajetórias melhoram bastante, desde quando eram simples trajetórias brutas, até agora, quando se tornaram trajetórias semânticas.

A Figura 20, é apresentado os pontos com enriquecimento semântico, após a execução dos algoritmos IB-SMoT e CB-SMoT com os melhores valores para seus parâmetros. Os pontos na cor vermelho, representam os pontos com enriquecimento semântico do algoritmo IB-SMoT. Os pontos na cor azul claro, representam os pontos com enriquecimento semântico do algoritmo CB-SMoT. Os pontos verdes são os que não possuem nenhum enriquecimento semântico.

Na Figura 21, foi escolhido um taxista aleatório e foi mostrado uma parte de sua trajetória com os pontos enriquecidos semanticamente e com os *stops* após a execução do

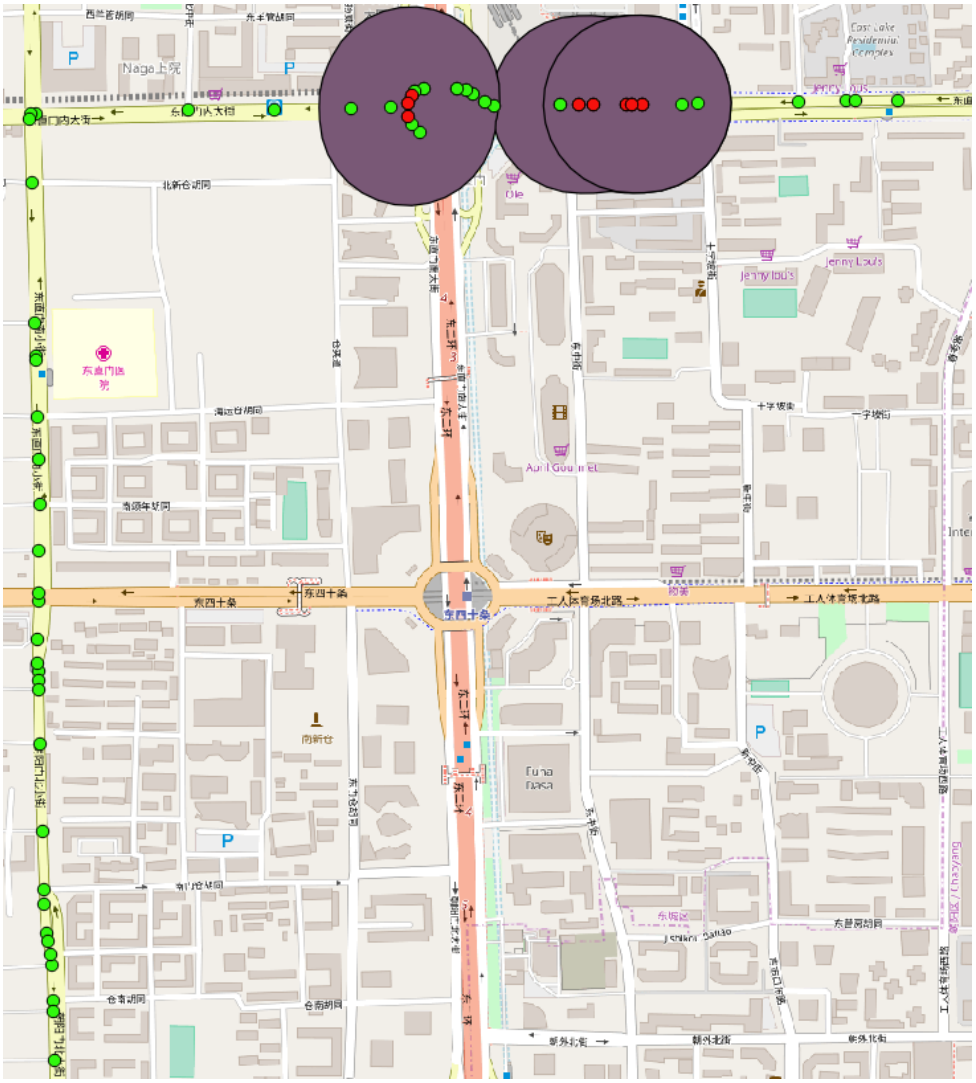
Figura 20 – Pontos com enriquecimento semântico (vermelho para o IB-SMoT e azul claro para o CB-SMoT) e pontos sem enriquecimento semântico (verde)



Fonte: Elaborado pelo autor

algoritmo do IB-SMoT. O taxista escolhido foi o que possui o *tid* sendo 9. O intervalo de tempo da trajetória começa as 08:20 e vai até as 09:20. É possível verificar que o algoritmo detectou 4 *stops*. O primeiro *stop* foi em um banco com o identificador 779, depois ele foi para um *fast food* com o identificador 719, depois voltou para o mesmo banco 779, em sequência, ele se dirigiu para um outro banco com o identificador 536. Por fim, o motorista seguiu em frente.

Figura 21 – Sub-trajetória enriquecida semanticamente de um taxista em um intervalo de tempo



Fonte: Elaborado pelo autor

8 CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma forma de realizar o enriquecimento semântico de trajetórias utilizando algoritmos para remover inconsistência de pontos geográficos, uma API para encontrar pontos de interesses de regiões e algoritmos para detecção de *stops*. Com o *framework* e a aplicação desenvolvidos neste trabalho, é possível, de forma fácil e prática, executar os algoritmos para encontrar os *stops*. Espera-se que outros trabalhos utilizem este *framework* e a aplicação e os adaptem para diversas situações e os melhorem.

Os resultados aqui encontrados, podem ser utilizados para determinar novos parâmetros para outros tipos de objetos em movimento e para outras bases de dados de outras regiões. Os resultados gerados pelo enriquecimento semântico podem ser utilizados para determinar os principais *stops* de uma determinada região, ou determinar os locais mais visitados por um determinado objeto em movimento.

O algoritmo para o *map matching* conseguiu fazer com que os pontos ficassem dentro da rede de ruas, isso melhorou a precisão e eficiência dos algoritmos que detectam os *stops*. A API conseguiu retornar diversos pontos de interesses, mas ainda é necessário modificar ou alterar para conseguir encontrar mais pontos. Os algoritmos de detecção de *stops* conseguiram realizar bem a sua função após uma boa definição dos parâmetros e a modificação para enriquecerem semanticamente os seus pontos. A análise dos parâmetros utilizados neste trabalho, é um exemplo de como determinar os melhores parâmetros para uma determinada base de dados e de objeto em movimento.

O Weka-STPM é uma ferramenta com um código muito antigo e a utilização apresentou diversas dificuldades. Uma das primeiras dificuldades ao iniciar o trabalho, foi com relação a como usar o Weka-STPM. Foi necessário baixar e importar para uma IDE. Depois foi necessário entender como o código funcionava e se os algoritmos estavam executando corretamente, pois não existe uma documentação para facilitar. O próximo passo foi entender em que e como os parâmetros influenciavam nos resultados. O próximo passo foi modificar o Weka-STPM para enriquecer semanticamente os pontos. E no final, a aplicação foi transformada em um projeto Spring Boot, removendo sua interface gráfica antiga e criando uma nova interface e uma estrutura. A estrutura do projeto foi completamente modificada para que outros usuários não tenham dificuldades em modificá-la.

8.1 Trabalhos Futuros

Este trabalho pode ser melhorado em diversos aspectos. Com relação a API de pontos de interesses, é necessário determinar uma nova API para realizar a captura desses pontos. A atual API não consegue capturar todos os pontos, mas talvez a API do Google Maps consiga um melhor resultado. Esse trabalho pode melhorar bastante se no futuro for realizado uma integração do algoritmo do *map matching*, da API de pontos de interesse e da nova modificação do Weka-STPM.

O Weka-STPM, até o momento, está apenas enriquecendo semanticamente os pontos e visualizando os valores, mas no futuro, espera-se que seja possível fazer o *framework* completo através dele, até a visualização dos dados usando a API do Google Maps. Um desafio para trabalhos futuros, é adaptar este trabalho para utilizar de *stream* de dados e definir novas estratégias para o *framework*.

A análise e eficiência dos resultados dos algoritmos é realizada de forma manual. É possível, através de estudos, definir uma maneira de tornar isso automático para o usuário. Não foi analisado como seria o resultado dos parâmetros utilizados nos algoritmos de enriquecimento semântico para os dados antes do *map matching*. É possível comparar isso e saber o ganho ou perda na qualidade dos dados usando o *map matching*.

REFERÊNCIAS

ALVARES, L. O.; BOGORNY, V.; KUIJPERS, B.; MACEDO, J. A. F. de; MOELANS, B.; VAISMAN, A. A model for enriching trajectories with semantic geographical information. In: ACM. **Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems**. Seattle, WA, USA, 2007. p. 22.

ALVARES, L. O.; BOGORNY, V.; KUIJPERS, B.; MOELANS, B.; FERN, J. A.; MACEDO, E.; PALMA, A. T. Towards semantic trajectory knowledge discovery. **Data Mining and Knowledge Discovery**, v. 12, 2007.

ALVARES, L. O.; PALMA, A.; OLIVEIRA, G.; BOGORNY, V. Weka-stpm: from trajectory samples to semantic trajectories. In: **Proceedings of the XI workshop de Software Livre, WSL**. Porto Alegre, RS: [s.n.], 2010. v. 10, p. 164–169.

CASSIANO, K. M. **Análise de Séries Temporais Usando Análise Espectral Singular (SSA) e Clusterização de Suas Componentes Baseada em Densidade**. Tese (Doutorado) — PUC-Rio, 2014.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Kdd**. Portland, Oregon: AAAI Press, 1996. v. 96, n. 34, p. 226–231.

FRANK, E.; HALL, M.; HOLMES, G.; KIRKBY, R.; PFAHRINGER, B.; WITTEN, I. H.; TRIGG, L. Weka-a machine learning workbench for data mining. In: **Data mining and knowledge discovery handbook**. New York City: Springer, 2009. p. 1269–1277.

LIU, L. X. Q.; LIU, M. L. Z. Map matching algorithm and its application. **Proceedings on Intelligent Systems and Knowledge Engineering (ISKE2007)**, 2007.

NEWSON, P.; KRUMM, J. Hidden markov map matching through noise and sparseness. In: ACM. **Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems**. Seattle, WA, USA: ACM, 2009. p. 336–343.

PALMA, A. L. T. **A clustering-based approach for discovering interesting places in trajectories**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, Instituto de Informática, Programa de Pós-Graduação em Computação, Porto Alegre, 2008.

PALMA, A. T.; BOGORNY, V.; KUIJPERS, B.; ALVARES, L. O. A clustering-based approach for discovering interesting places in trajectories. In: ACM. **Proceedings of the 2008 ACM symposium on Applied computing**. Fortaleza, Ceara, Brazil: ACM, 2008. p. 863–868.

PARKINSON, B. W. **Global positioning system: Theory and applications, Volume II**. Reston: AIAA, 1996.

SPACCAPIETRA, S.; PARENT, C.; DAMIANI, M. L.; MACEDO, J. A. de; PORTO, F.; VANGENOT, C. A conceptual view on trajectories. **Data & knowledge engineering**, Elsevier, v. 65, n. 1, p. 126–146, 2008.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining, (First Edition)**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN 0321321367.

TRAN, T. N.; DRAB, K.; DASZYKOWSKI, M. Revised dbscan algorithm to cluster data with dense adjacent clusters. **Chemometrics and Intelligent Laboratory Systems**, Elsevier, v. 120, p. 92–96, 2013.

WIKIPéDIA. **Teoria dos grafos — Wikipédia, a enciclopédia livre**. 2018. Online. Disponível em: https://pt.wikipedia.org/w/index.php?title=Teoria_dos_grafos&oldid=51770047. Acesso em: 10 abr. 2018.

YAN, Z.; CHAKRABORTY, D.; PARENT, C.; SPACCAPIETRA, S.; ABERER, K. Semitri: a framework for semantic annotation of heterogeneous trajectories. In: ACM. **Proceedings of the 14th international conference on extending database technology**. Uppsala, Sweden: ACM, 2011. p. 259–270.

YAN, Z.; GIATRAKOS, N.; KATSIKAROS, V.; PELEKIS, N.; THEODORIDIS, Y. Setrastream: Semantic-aware trajectory construction over streaming movement data. In: PFOSER, D.; TAO, Y.; MOURATIDIS, K.; NASCIMENTO, M. A.; MOKBEL, M.; SHEKHAR, S.; HUANG, Y. (Ed.). **Advances in Spatial and Temporal Databases**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 367–385. ISBN 978-3-642-22922-0.

ZHENG, Y. **T-Drive trajectory data sample**. 2011. Online. Disponível em: https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/. Acesso em: 23 mai. 2018.