UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MESTRADO ACADÊMICO EM COMPUTAÇÃO

WILKEN CHARLES DANTAS DE MELO

USO DE MODELOS DE LINGUAGEM PARA A AVALIAÇÃO DE SIMILARIDADE ESPACIAL ENTRE TRAJETÓRIAS

QUIXADÁ

2024

WILKEN CHARLES DANTAS DE MELO

USO DE MODELOS DE LINGUAGEM PARA A AVALIAÇÃO DE SIMILARIDADE ESPACIAL ENTRE TRAJETÓRIAS

Dissertação apresentada ao curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação da Universidade Federal do Ceará (UFC) - Campus de Quixadá, como requisito parcial para a obtenção do título de mestre em computação. Área de concentração: Ciência da Computação

Orientador: Prof. Dr. Regis Pires Magalhães

Coorientadora: Profa. Dra. Lívia Almada Cruz

QUIXADÁ

2024

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

M486u Melo, Wilken Charles Dantas de.
Uso de modelos de linguagem para a avaliação de similaridade espacial entre trajetórias / Wilken Charles Dantas de Melo. – 2024.
89 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2024.
Orientação: Prof. Dr. Regis Pires Magalhães.
Coorientação: Profa. Dra. Lívia Almada Cruz.

1. Trajetórias Espaciais. 2. Modelos de Linguagem. 3. Similaridade entre Trajetórias. 4. Embeddings de Trajetórias. 5. Processamento de Linguagem Natural. I. Título.

CDD 005

WILKEN CHARLES DANTAS DE MELO

USO DE MODELOS DE LINGUAGEM PARA A AVALIAÇÃO DE SIMILARIDADE ESPACIAL ENTRE TRAJETÓRIAS

Dissertação apresentada ao curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação da Universidade Federal do Ceará (UFC) - Campus de Quixadá, como requisito parcial para a obtenção do título de mestre em computação. Área de concentração: Ciência da Computação

Aprovada em: 11/04/2024

BANCA EXAMINADORA

Prof. Dr. Regis Pires Magalhães (Orientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Lívia Almada Cruz (Coorientadora)
Universidade Federal do Ceará (UFC)

Dr. Francesco Lettich
ISTI - CNR, Pisa, Itália

Dr. Vinícius Cezar Monteiro de Lira
Universidade Federal do Ceará (UFC)

Primeiramente, a Deus, por estar sempre ao meu lado. Em segundo lugar, aos meus familiares e amigos, por todo o apoio na minha trajetória de formação acadêmica.

AGRADECIMENTOS

A Deus, por ter me guiado e dado forças nos momentos mais difíceis da minha vida, permitindo sempre que eu pudesse prosseguir.

Aos meus pais, Noé e Hozana, por todo amor e incentivo dados durante toda minha vida. Agradeço o amor de vocês, que foi fundamental para minha jornada até aqui.

À minha esposa Luciana e à minha filha Maria Cecília, pelo amor, companheirismo e compreensão nos momentos de ausência.

Aos meus irmãos, Paula, Paulo, Flávio, Fábio e Shyrliana, pela parceria de vida e pelo apoio neste projeto.

A minha sogra, Tânia, por sua fé e orações direcionadas a mim.

Ao meu amigo, Zilzon Lima, por sua amizade simples e verdadeira. Que Deus o abençoe e conceda sua recuperação plena.

Aos meus orientadores, Prof. Dr. Regis Pires e Profa. Dra. Lívia Almada, pelos ensinamentos, atenção, incentivo e, sobretudo, por compreenderem minhas condições como aluno, servidor e pai. Regis, sou muito grato pelos conselhos valiosos e por sempre ser um orientador presente.

À Profa. Dra. Ticiana Linhares por todos os ensinamentos passados durante os experimentos deste trabalho. Obrigado por sempre estar presente e disposta a ajudar.

Ao pesquisador Dr. Francesco Lettich pelos valiosos direcionamentos quanto ao escopo desta pesquisa. Agradeço também pela imensa contribuição na elaboração do nosso artigo científico.

Aos amigos do grupo de pesquisa "Trajetórias DL", Matheus Cordeiro e Guilherme Sales, pelo apoio no desenvolvimento desta pesquisa e pela amizade construída.

Aos amigos que fiz durante minha jornada no PCOMP, sou grato ao Humberto e Lucas pelo apoio nos momentos difíceis das disciplinas que superamos juntos.

Por fim, agradeço a todos os meus colegas de trabalho da UFC Campus Russas, em especial aos meus companheiros do DTIC, pelo apoio constante e pela colaboração ao longo desta minha jornada acadêmica.

“Nós só podemos ver um pouco do futuro, mas o suficiente para perceber que há muito a fazer.”

(Alan Turing)

RESUMO

A proliferação de dispositivos eletrônicos capazes de fornecer informações geoespaciais, tais como celulares, automóveis, dispositivos pessoais, dentre outros, tem levado a um aumento sem precedente na geração de dados de trajetórias. Esses dados são cruciais para vários domínios do aprendizado de máquina, especialmente a análise de mobilidade. O presente trabalho tem como foco um problema inerente deste domínio, a avaliação de similaridade entre trajetórias. Pesquisas recentes buscam transformar trajetórias em *embeddings*, representação vetorial compacta que consegue capturar eficientemente as características dos trajetos. A ideia fundamental é que trajetórias similares tenham *embeddings* próximos em seu espaço vetorial. Embora existam vários métodos de aprendizado profundo que geram *embeddings* de trajetórias, este trabalho se concentra naqueles que primeiro discretizam as trajetórias, usando uma grade uniforme, e depois geram as incorporações. Neste contexto, considerou-se como referência a abordagem *t2vec*. Um campo em paralelo é o Processamento de Linguagem Natural (PLN), o qual envolve a conversão de imensos corpus textuais em vetores numéricos que conseguem capturar uma variedade de contextos semânticos. Os avanços em modelos de linguagem mais robustos, *Large Language Models (LLMs)*, tais como *BERT* e *GPT*, demonstram sua notável capacidade de imitar o raciocínio humano: isto sugere uma alternativa potencial para capturar padrões espaço-temporais de mobilidade, uma direção de investigação relativamente inexplorada. O presente trabalho investiga se modelos de linguagem podem ser reutilizados para produzir *embeddings* de alta qualidade para trajetórias. Em particular, argumenta-se que trajetórias adequadamente discretizadas podem ser tratadas como palavras ou sentenças, permitindo, assim, que os modelos de linguagem identifiquem padrões e relações nesses dados. Na avaliação experimental, considerou-se dois conjuntos públicos de dados de trajetórias (Porto e T-drive). Em seguida, o desempenho de quatro modelos de linguagem bem estabelecidos (*Word2Vec*, *Doc2Vec*, *BERT* e *SBERT*) foi comparado com o *t2vec*. Além disso, também foram considerados métodos clássicos de similaridade para fornecer uma comparação mais abrangente. Os resultados obtidos indicam que os modelos de linguagem, quando treinados em conjuntos de dados com trajetórias densas, podem gerar *embeddings* de maior qualidade do que *t2vec*, destacando assim o forte potencial dessas abordagens.

Palavras-chave: Trajetórias Espaciais; Modelos de Linguagem; Similaridade entre Trajetórias; *Embeddings* de Trajetórias; Processamento de Linguagem Natural.

ABSTRACT

The proliferation of electronic devices capable of providing geospatial information, such as cell phones, automobiles, personal devices, among others, has led to an unprecedented increase in trajectory data generation. This data is crucial for various machine learning domains, especially mobility analysis. The present work focuses on an inherent problem in this domain, the evaluation of trajectory similarity. Recent research seeks to transform trajectories into embeddings, compact vector representations that can efficiently capture the characteristics of paths. The fundamental idea is that similar trajectories have close embeddings in their vector space. Although there are several deep learning methods that generate trajectory embeddings, this work focuses on those that first discretize trajectories, using a uniform grid, and then generate the embeddings. In this context, the t2vec approach was considered as a reference. A parallel field is Natural Language Processing (NLP), which involves converting vast textual corpora into numerical vectors that can capture a variety of semantic contexts. Advances in more robust language models, such as BERT and GPT, demonstrate their remarkable ability to mimic human reasoning: this suggests a potential alternative for capturing spatiotemporal mobility patterns, a relatively unexplored research direction. The present work investigates whether language models can be repurposed to produce high-quality embeddings for trajectories. In particular, it is argued that adequately discretized trajectories can be treated as words or sentences, thus allowing language models to identify patterns and relationships in this data. In the experimental evaluation, two public trajectory datasets (Porto and T-drive) were considered. Then, the performance of four well-established language models (Word2Vec, Doc2Vec, BERT, and SBERT) was compared with t2vec. Additionally, classical similarity methods were also considered to provide a more comprehensive comparison. The results indicate that language models, when trained on datasets with dense trajectories, can generate higher-quality embeddings than t2vec, thus highlighting the strong potential of these approaches.

Keywords: Spatial Trajectories. Language Models. Trajectory Similarity. Trajectory Embeddings. Natural Language Processing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

GPS	*Global Positioning System*
RNN	*Recurrent Neural Networks*
EST	*External Sensor Trajectory*
POI	*Points of Interest*
RNA	*Redes Neurais Artificiais*
RN	*Redes Neurais*
RNB	*Redes Neurais Biológicas*
PLN	*Prcessamento de Linguagem Natural*
IA	*Inteligência Artificia*
LLMs	*Large Language Models*
CBOW	*Continuous Bag of Words*
PV-DM	*Distributed Memory version of Paragraph Vector*
PV-DBOW	*Distributed Bag of Words version of Paragraph Vector*
BERT	*Bidirectional Encoder Representations from Transformers*
DTW	*Dynamic Time Warping*
ED	*Edit Distance*
EDR	*Edit Distance on Real Sequence*
LCSS	*Longest Common Subsequence*
LSTM	*Long Short-Term Memory*
SBERT	*Sentence BERT*
GRU	*Gated Recurrent Unit*
GPT	*Generative Pre-trained Transformers*
LlaMa2	*Large Language Model Meta AI 2*

LISTA DE SÍMBOLOS

$\mathbb{T}$	Conjunto de todas as trajetórias (*dataset*)
T	Trajetória espacial (pontos de GPS)
T'	Trajetória discretizada (células da grid)
$\|T\|$ ($\|T'\|$)	Comprimento da trajetória T (T')
p_i	i-éssimo ponto de uma trajetória T
C	Grade uniforme
c	Célula da grade C
v^T	*Embedding* da trajetória T
sr	Taxa média de captação dos pontos
R	Rota real de uma trajetória T
V	Vocabulário de células (*Hot cells*)
mr	Ranque médio (*Mean Ranking*)
ε	Intervalo de subcusto para os métodos clássicos

SUMÁRIO

1 INTRODUÇÃO

A diversidade de dispositivos eletrônicos dotados de mecanismos de geoposicionamento, *Global Positioning System* (GPS), presentes na vida cotidiana, como celulares, câmeras e sensores, gera uma vasta quantidade de dados geoespaciais diariamente. O uso desses dispositivos fez o volume desta classe de informação tomar proporções gigantescas nos últimos anos, consequentemente, impulsionando pesquisas relevantes quanto ao processamento e análise desta categoria de dados. Inúmeras são as aplicações que usam dados geoespaciais como base da informação, algumas são citadas na resolução de problemas de mobilidade humana (Cao *et al.*, 2020; Wang *et al.*, 2013), em análises meteorológicas (Shuncheng *et al.*, 2019) e, até mesmo, na identificação de rotas migratórias e fenômenos naturais (Li *et al.*, 2018b). Essa gama de aplicabilidades reforça o quanto dados de trajetórias podem ser importantes para a tomada de decisão, podendo impactar positivamente na vida humana.

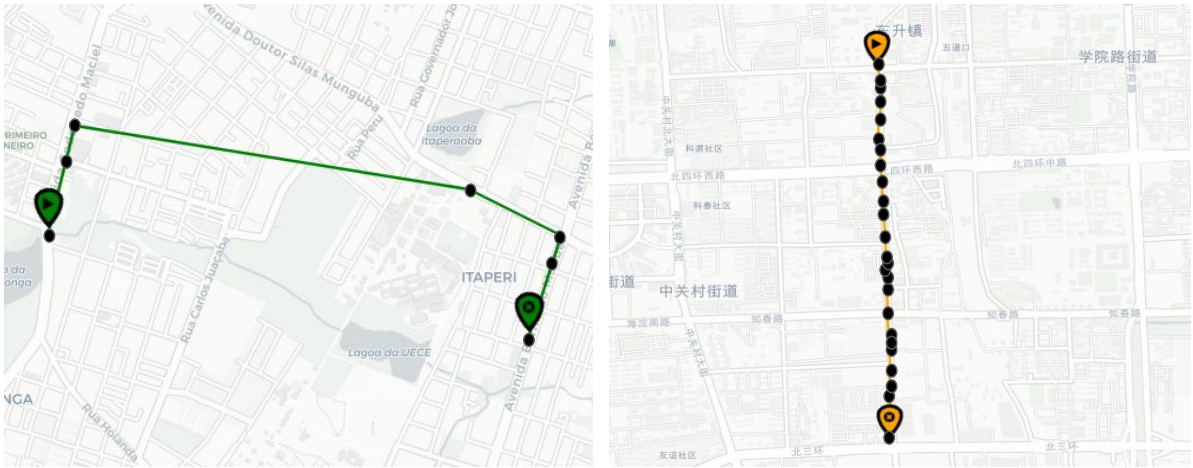
Segundo Zheng (2015), uma trajetória espacial pode ser definida como um conjunto de pontos gerados no espaço geográfico por um objeto móvel, seguindo uma ordem cronológica na distribuição espaço-temporal dos seus pontos, onde cada ponto carrega a informação geoespacial (longitude, latitude) mais um carimbo de tempo (*timestamp*). A maneira como os movimentos de um objeto são capturados está diretamente ligada à forma como ele é observado: ativa ou passiva. Na forma ativa, geralmente, os objetos móveis (e.g., celulares, veículos, embarcações) possuem dispositivos de GPS integrados que conseguem informar a localização exata do objeto em curtos intervalos de tempo, proporcionando o registro dos dados em frações de segundos. Na forma passiva, os movimentos são capturados por dispositivos externos (e.g., sensores ou câmeras) ao objeto observado. Neste modo de captação, as trajetórias são geradas com baixa taxa amostral de pontos e possuem particularidades específicas impostas por este modo de captação.

Em Cruz *et al.* (2019) a predição do próximo sensor é abordada em trajetórias oriundas de sensores externos, denominadas pelos autores de *External Sensor Trajectory (EST)*, trajetórias coletadas passivamente. No artigo, as trajetórias utilizadas são provenientes de um conjunto de dados no qual os trajetos são coletados por sensores de vigilância de trânsito em uma rede de rodovias. Estes dados de trajetórias têm suas particularidades e são bastante desafiadores, pois os sensores envolvidos na captação encontram-se, geralmente, bastante afastados uns dos outros, causando uma maior esparsidade (distanciamento) entre os dados coletados. Já em Li *et al.* (2018b), o cálculo da similaridade espacial entre trajetórias é dado por meio de trajetórias

coletadas ativamente, usando dispositivos de *GPS* integrados aos automóveis de uma rede de táxis das cidades de Porto (Portugal) e Harbin (China). Nesses conjuntos de dados, as trajetórias possuem taxas amostrais mais altas e são mais propícias a ruídos e interferências do ambiente (e.g., prédios, montanhas, viadutos, etc.).

Independente da abordagem de coleta, dados de trajetórias são considerados complexos devido às suas características intrínsecas, requerendo uma maior atenção no seu processo de captação, armazenamento e manuseio (Cao *et al.*, 2020; Li *et al.*, 2018b). Com intuito de elucidar, na Figura 1 é ilustrada a questão da esparsidade dos dados, fazendo um comparativo entre duas trajetórias reais, de mesmo tamanho (2,5 km), coletadas nas duas formas: passiva e ativa. No lado esquerdo (a) da figura, apresenta-se uma trajetória obtida de forma passiva (7 pontos). Já no lado direito (b), tem-se uma trajetória de mesmo comprimento, porém obtida de forma ativa (34 pontos). Neste exemplo, a diferença na densidade de pontos entre esses dois tipos de trajetórias torna-se bastante evidente.

Figura 1 – Trajetórias por coleta passiva (lado esquerdo) e coleta ativa (lado direito).



Fonte: Elaborado pelo autor (2024)

O presente trabalho aborda o cálculo da similaridade espacial entre trajetórias densas ou esparsas, o qual envolve determinar a similaridade entre duas trajetórias quaisquer, baseado em suas características espaciais. Na literatura, foram propostos diversos métodos para lidar com esse problema, conhecidos como métodos clássicos. Entre eles, destacam-se a *Dynamic Time Warping (DTW)* (Kruskal, 1983), *Longest Common Sub-Sequence (LCSS)* (Shuncheng *et al.*, 2019) e a *Edit Distance on Real Sequences (EDR)* (Levenshtein, 1965). O objetivo principal destes métodos é fazer o alinhamento entre os pontos das duas trajetórias de forma

otimizada, usando a programação dinâmica. Especificamente, pontos relativamente próximos são combinados entre si visando encontrar o alinhamento mais eficiente (ou menos custoso) entre as duas trajetórias. Então, a medida de similaridade final é baseada no custo total (ou distância) calculada a partir dessas combinações. No entanto, a principal desvantagem desses métodos é o alto custo computacional, uma vez que escalam quadraticamente com o comprimento das trajetórias.

Trabalhos recentes (Wang *et al.*, 2020; Li *et al.*, 2018b; Fang *et al.*, 2022; Fu; Lee, 2020; Zhang *et al.*, 2019) exploraram o *aprendizado de representação* como uma alternativa para enfrentar esses desafios computacionais. Essa abordagem busca extrair características valiosas e de baixa dimensão, em algum espaço latente, a partir de dados complexos, por meio do treinamento de modelos baseados em aprendizado profundo. Assim, um modelo treinado pode ser usado para converter trajetórias em vetores (*embeddings*) que, em seguida, podem ser usados em tarefas específicas, como o cálculo da similaridade espacial entre trajetórias: em que, trajetórias semelhantes devem ter vetores também semelhantes (próximos) no seu espaço latente.

A maioria das abordagens que visam a similaridade entre trajetória frequentemente usam modelos sofisticados baseados em *Recurrent Neural Networks* (RNN), *transformers* ou mecanismos de atenção que requerem alguma discretização inicial necessária para o treino dos modelos. Tal discretização pode ser obtida por meio do particionamento do espaço, usando uma grade, e.g., (Li *et al.*, 2018b), ou uma rede de segmentos, e.g., (Fang *et al.*, 2022). No presente trabalho, foi adotada a discretização baseada em uma grade uniforme de células. Enquanto a discretização baseada em uma rede de segmentos pode captar muito bem a rota padrão (*underlying route*) dos trajetos na rede, provendo mais qualidade aos *embeddings*, abordagens baseadas em grades possuem outras vantagens, i.e., elas são (1) independentes da rede ruas, (2) aplicáveis a entidades que não estão diretamente ligadas às ruas, como pedestres, e (3) evitam a etapa demorada de pré-processamento, potencialmente propensa a erros, para realizar o mapeamento entre todos os pontos e seus respectivos segmentos (*map matching*). Finalmente, nota-se que as abordagens existentes focam em diferentes tipos de similaridade entre trajetórias, i.e., espacial, espaço-temporal ou multi-modal, cada uma requerendo uma função de perda específica durante o treino. Neste trabalho, focou-se no problema de avaliar a *similaridade espacial* entre trajetórias.

Uma área aparentemente distante é o Processamento de Linguagem Natural (PLN), em que os elementos de grandes corpus textuais são transformados em vetores numéricos,

conseguindo incorporar diversos contextos semânticos. Com os avanços recentes no PLN, especialmente com os *Large Language Models* (LLMs) como o *BERT* e *GPT*, ficou demonstrado que esses modelos conseguem desenvolver habilidades de raciocínio semelhantes às dos seres humanos.

Ao traçar paralelos entre a natureza sequencial da linguagem e de trajetos, é possível imaginar o uso de modernos modelos de linguagem, como o *BERT*, para criar *embeddings* de trajetórias com alta qualidade. A ideia central é tratar trajetórias discretizadas como palavras ou sentenças em textos, permitindo aos modelos de linguagem identificar padrões e relações em dados de objetos móveis. Na verdade, esta intuição se alinha com pesquisas recentes que usam modelos de PLN para o aprendizado de representação de trajetória. Por exemplo, o *Habit2vec* (Cao *et al.*, 2020) modela hábitos de vida a partir de trajetórias em *embeddings*. *TraceBERT* (Crivellari *et al.*, 2022) infere observações espaciais ausentes de objetos em movimento com base em locais visitados anteriormente e Cruz *et al.* (2022) usa métodos de PLN para codificar dados de sensores externos de uma rede rodoviária para gerar incorporações e prever a próxima localização de veículos que passam por esses sensores.

O reaproveitamento de modelos de linguagem, especialmente modelos grandes como o *BERT*, para gerar *embeddings* de trajetórias acarreta desafios específicos. Uma preocupação primária é a necessidade de discretizar as trajetórias em *tokens*, elementos fundamentais para o vocabulário dos modelos de linguagem. Existem diversos métodos com esse propósito na literatura, incluindo grade de células, pontos de interesse, sensores de trânsito e rede de segmentos. A utilização da discretização baseada em grade é vantajosa devido à facilitação de um vocabulário muito menor, simplificando o processo de aprendizagem e reduzindo a complexidade de treinamento dos modelos. Em segundo lugar, treinar grandes modelos como o *BERT* a partir do zero exige o desenvolvimento de um vocabulário que se alinhe com a abordagem de discretização selecionada. Este processo requer dados de trajetória suficientes para aprender representações ideais para cada *token* do vocabulário. Além disso, requer a inicialização e o treinamento de inúmeros parâmetros, exigindo, portanto, recursos computacionais consideráveis.

1.1 Questões de Pesquisa

Dada a eficácia dos modelos de PNL, em particular dos *LLMs*, na aprendizagem de padrões sequenciais, o presente trabalho pretende reaproveitar estes modelos para aprender características espaciais essenciais das trajetórias visando obter incorporações de alta qualidade

que podem potencialmente competir com o estado da arte atual. Dessa forma, as questões de pesquisa abordadas neste trabalho são as seguintes:

QP1. Modelos de linguagem são capazes de gerar *embeddings* de alta qualidade que representam efetivamente tanto trajetórias densas quanto esparsas, refletindo a similaridade espacial entre elas?

QP2. Modelos de linguagem são escaláveis à medida que se amplia o espaço de busca no qual as representações (*embeddings*) são comparadas?

1.2 Estrutura do Trabalho

Este trabalho é composto pela seguinte estruturação. O Capítulo 1 apresenta a parte introdutória da dissertação. O Capítulo 2 mostra a fundamentação teórica necessária para o entendimento do problema. O Capítulo 3 apresenta e compara os trabalhos relacionados. O Capítulo 4 expõe os procedimentos metodológicos. O Capítulo 5 explica os experimentos e resultados preliminares. Por fim, o Capítulo 6 apresenta as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados conceitos preliminares sobre trajetórias e as principais notações utilizadas neste trabalho. Para uma melhor compreensão, trajetórias são definidas como sequências de pontos geoespaciais cronologicamente identificados que descrevem o histórico de movimento dos objetos no espaço-tempo, tal como: pessoas, animais, veículos e, até mesmo, fenômenos naturais.

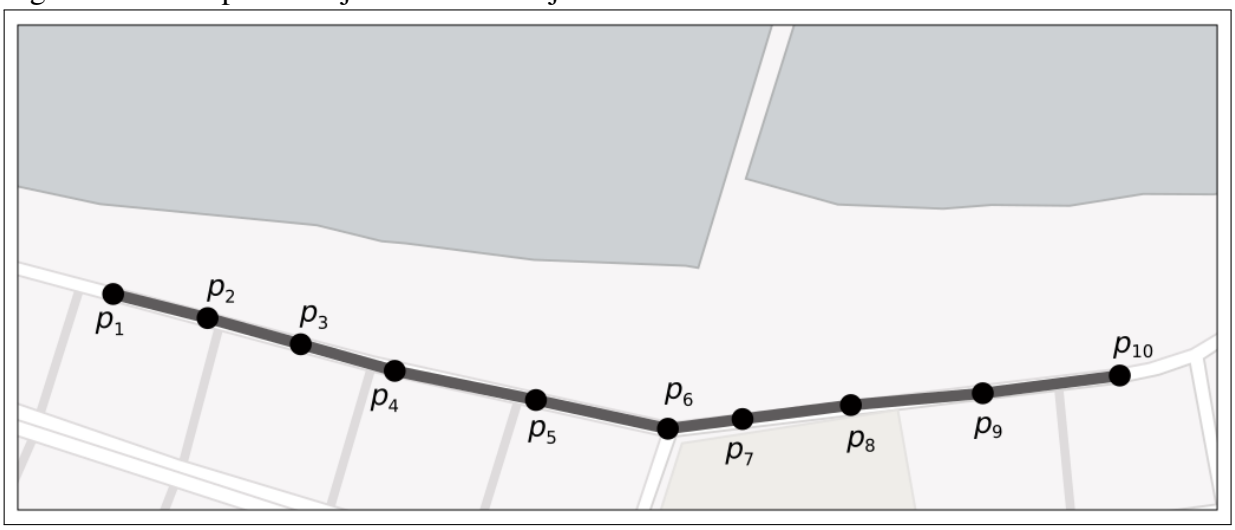
De acordo com Shuncheng *et al.* (2019), teoricamente, uma trajetória precisa ter seus pontos registrados continuamente, ou seja, deve ser representada por uma função matemática contínua no tempo, dado que o movimento do objeto em questão é contínuo por natureza. Porém, na prática, o registro de localizações contínuas para um objeto em movimento nem sempre é possível (ou viável), visto que os dispositivos de geoposicionamento (e.g., sistemas de *GPS*, câmeras de trânsito, sensores, dentre outros) só conseguem captar a posição atual do objeto móvel de maneira periódica, ou seja, em momentos exatos no espaço-tempo.

2.1 Conceitos Preliminares e Definição do Problema

Devido às limitações, mencionadas anteriormente, intrínsecas aos dispositivos responsáveis pela coleta e armazenamento dos dados geoespaciais, as localizações geográficas dos objetos móveis de natureza contínua são coletadas e armazenadas de maneira discreta, ou seja, em intervalos de tempo bem definidos. A Figura 2 ilustra um exemplo de trajetória discreta contendo 10 pontos (amostras). Desta forma, é comum tratarem trajetórias de objetos móveis coletadas por esses dispositivos como as trajetórias reais dos objetos, apesar desas trajetórias serem apenas uma representação mais próxima possível dos valores reais dos trajetos (Zheng, 2015). Com base nisso, a definição do problema e as principais notações utilizadas neste trabalho são introduzidas:

Definição 1 *(Trajetória T)*. Uma trajetória espacial T, gerada por um objeto móvel no espaço-tempo, é definida como uma série de localizações (ou amostras) cronologicamente ordenadas, tal como: $T = \langle p_1, p_2, \ldots, p_n \rangle$. Onde cada ponto $p_i = (x_i, y_i, t_i)$ representa a i-ésima posição do objeto no espaço e no tempo, com x_i e y_i indicando sua localização geográfica, e t_i o carimbo de data/hora em que a amostra foi registrada.

Figura 2 – Exemplo de trajetória de um objeto móvel



Fonte: Adaptada de GOMES (2018)

Este trabalho se concentra na transformação de trajetória em *embeddings*, por meio de modelos de linguagem, usando a discretização baseada em grade uniforme. Assim, o conceito de particionamento de uma área geográfica baseado em uma grade uniforme é dado a seguir.

Definição 2 *(Particionamento de área geográfica baseado em grade uniforme).* Seja G o retângulo mínimo de borda (RMB) que delimita a área geográfica de interesse. Denota-se tal RMB por $G = (x_a^G, y_a^G, x_b^G, y_b^G)$, onde (x_a^G, y_a^G) e (x_b^G, y_b^G) representam respectivamente os cantos inferior esquerdo e superior direito do RMB. Então, G é particionado de acordo com uma grade uniforme C de NxM células, onde cada célula tem largura W e altura H, de modo que a célula c_{ij} cobre a seguinte região:

$$\left(x_a^G + i \cdot W, x_a^G + (i+1) \cdot W, y_a^G + j \cdot H, y_a^G + (j+1) \cdot H\right).$$

Para garantir que a grade C cobre a área G, constantes N, M, W, e H foram escolhidas de modo que $x_a^G + N \cdot W > x_b^G$ e $y_a^G + M \cdot H \geq y_b^G$ sejam válidos. Para cada célula $c \in C$ foi associado um identificador inteiro, de forma que esta associação deve impor uma ordenação entre os índices das células.

Desta forma, é possível definir como as trajetórias são discretizadas usando a grade uniforme C.

Definição 3 *(Discretização de trajetória baseada em grade uniforme).* Dado uma trajetória $T = \langle p_1, \ldots, p_j \rangle$ e uma grade uniforme C, sua discretização consiste em converter cada ponto $p \in T$ em um identificador de célula $c \in C$, de maneira que c contenha a localização de p. O resultado é uma trajetória discretizada $T' = \langle c_1, \ldots, c_m \rangle$, onde $m \leq j$.

É importante notar que umas das consequências da discretização é a transformação da trajetória em uma sequência de *tokens* processáveis por modelos de PLN. Além disso, a trajetória discretizada pode ser mais curta que a original. De fato, neste trabalho pontos consecutivos que caem dentro da mesma célula são interpretados com um único identificador de célula em T', o que acaba por encurtar a trajetória. Assim, a noção de vetorização (*embedding*) a partir de uma trajetória discretizada pode ser dada.

Definição 4 *(Geração do embedding da trajetória).* O processo de geração do *embedding* de uma trajetória envolve a transformação de uma trajetória discretizada $T' = \langle c_1, \ldots, c_m \rangle$ em uma representação vetorial $v^T \in \mathbb{R}^d$, onde v^T representa o *embedding* da trajetória original T e $\mathbb{R}^d$ denota o espaço d-dimensional desse *embedding*. Esta transformação é alcançada através da função $f : C^* \to \mathbb{R}^d$, onde C^* é o conjunto de todas as sequências de células possíveis obtidas a partir da grade uniforme C. Assim, a representação vetorial v^T deve encapsular todas as características espaciais da trajetória original T.

Com base nas definições preliminares, torna-se possível apresentar o problema abordado neste trabalho.

Definição 5 *(Definição do problema).* O problema consiste em gerar representações vetoriais (*embeddings*) com alta qualidade para serem usadas no cálculo de similaridade espacial entre trajetórias. Sendo $\mathbb{T}$ o conjunto de todas as trajetórias, $\mathbb{T}'$ o conjunto correspondente das trajetórias discretizadas e f a função de geração de *embeddings* para as trajetórias em $\mathbb{T}'$. Então, para qualquer trajetória $T \in \mathbb{T}$, tem-se T' como sua trajetória discretizada correspondente em $\mathbb{T}'$, sendo $v^T = f(T')$ o respectivo *embedding* de T gerado pela função f. Idealmente, o ranking produzido medindo-se as distâncias entre v^T e todos os outros *embeddings* em $\mathbb{T}'$ deveria ser equivalente ou o mais próximo possível do ranking produzido medindo-se a similaridade espacial entre T e as demais trajetórias em $\mathbb{T}$, via alguma função d de similaridade espacial ideal para trajetórias.

Em outras palavras, o objetivo deste trabalho consiste em validar se os *embeddings* gerados pela função f conseguem preservar as relações de similaridade espacial inerentes às trajetórias originais, de modo que as avaliações de similaridade baseadas em *embeddings* se alinhem àquelas derivadas diretamente das propriedades espaciais das trajetórias.

Além disso, a fim de melhorar o entendimento do problema e das funções d adotadas, as seguintes notações são usadas no decorrer deste trabalho:

1. *Topo(T)*. Dado uma trajetória $T = \langle p_1, p_2, p_3, \ldots, p_n \rangle$, topo será sempre seu primeiro ponto, $Topo(T) = p1$.
2. *Resto(T)*: Para a mesma trajetória $T = \langle p_1, p_2, p_3, \ldots, p_n \rangle$, o resto será composto por todos os pontos exceto o ponto inicial Topo(T), assim, $Resto(T) = \langle p_2, p_3, \ldots, p_n \rangle$.
3. *Comprimento(T)*: Representa o tamanho de uma trajetória T com base no seu número de pontos (*GPS*) ou seu número de células (na grade *C*). Por exemplo, a trajetória original $T = \langle p_1, p_2, p_3, \ldots, p_n \rangle$ tem $Comprimento(T) = n$. Já a trajetória discretizada $T' = \langle c_1, c_4, c_7, c_9 \rangle$ tem $Comprimento(T') = 4$.

2.2 Trajetórias Densas

Trajetórias com alta densidade de informação ocorrem principalmente quando os dispositivos, responsáveis pela geração de seus pontos, conseguem empregar taxas elevadas de amostragem na criação destes. Para Zheng (2015), trajetórias com este perfil, mesmo possuindo elevados volumes de dados, são as que melhor conseguem representar o histórico de movimento dos objetos móveis envolvidos.

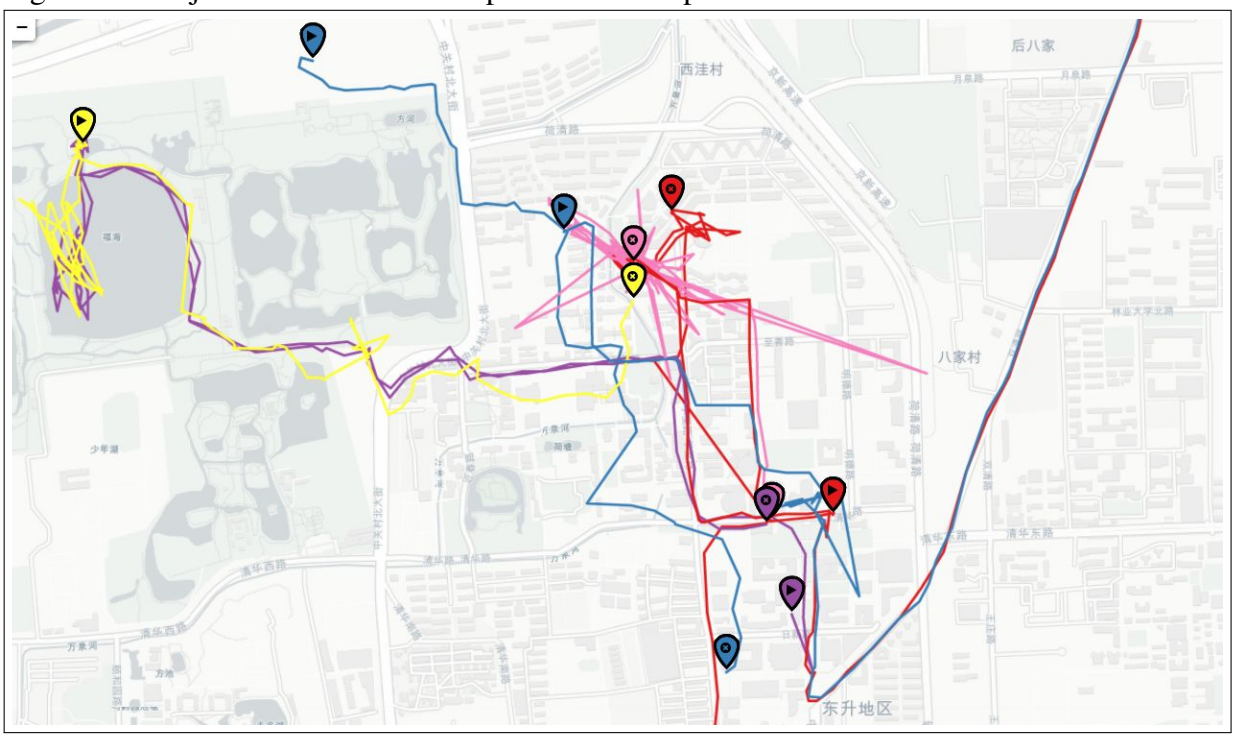
Devido ao surgimento de novas tecnologias, principalmente dos dispositivos dotados de *GPS*, houve um aumento sem precedentes no volume dados que conseguem descrever precisamente o movimento histórico de objetos móveis, tais como pessoas, automóveis e celulares. Shuncheng *et al.* (2019) ressaltam que dados com essas características são essenciais para análises em larga escala, e diversas pesquisas têm se utilizado deles em uma variedade de contextos. Por exemplo, alguns trabalhos incluem dados de trajetórias densas na elaboração de estruturas de indexação para trajetórias (Chen *et al.*, 2005; Cudre-Mauroux *et al.*, 2010), na pesquisa de alta desempenho para encontrar trajetórias mais similares (Wang *et al.*, 2013; Li *et al.*, 2018b; Zhang *et al.*, 2019), nos métodos de mineração de dados para encontrar padrões de trajetórias e na

predição das condições de tráfego (Lee *et al.*, 2007; Li *et al.*, 2010).

Trajetórias de perfil denso envolvem processos específicos de geração e captação dos seus pontos, em Zheng (2015) duas categorias principais são citadas: *mobilidade de pessoas* e *mobilidade de veículos de transporte*. Na primeira, as trajetórias são criadas com base nos movimentos reais dos objetos móveis (pessoas), usando-se geralmente aparelhos celulares equipados com *GPS*, onde a captação dos pontos ocorre de forma ativa por longos períodos de tempos e com altas taxas de amostrais. Na segunda categoria, as trajetórias são formadas por variados tipos de veículos de transporte (e.g., carros, motos, embarcações, etc.), em que esses objetos são equipados com dispositivos de geolocalização (*GPS*), conseguindo gerar as informações espaço-temporais também de forma ativa e com elevadas taxas amostrais.

Diversos são os cenários nos quais as trajetórias densas podem ser usadas, envolvendo os mais variados domínios e aplicações de uso. Assim, a maneira como estes dados se comportam requer um pouco a mais de atenção no seu pré-processamento e manuseio, pois a probabilidade da inserção de ruídos nessa categoria de dados é muito maior que nas trajetórias consideradas esparsas. Para uma melhor compreensão de como estas trajetórias se comportam, na Figura 3 é ilustrado uma amostra real de trajetórias densas provenientes da coleta ativa por *GPS*, envolvendo as duas categorias anteriormente citadas, *mobilidade de pessoas* (trajetórias nas cores lilás e amarela) e *mobilidade de veículos de transporte* (trajetórias nas demais cores).

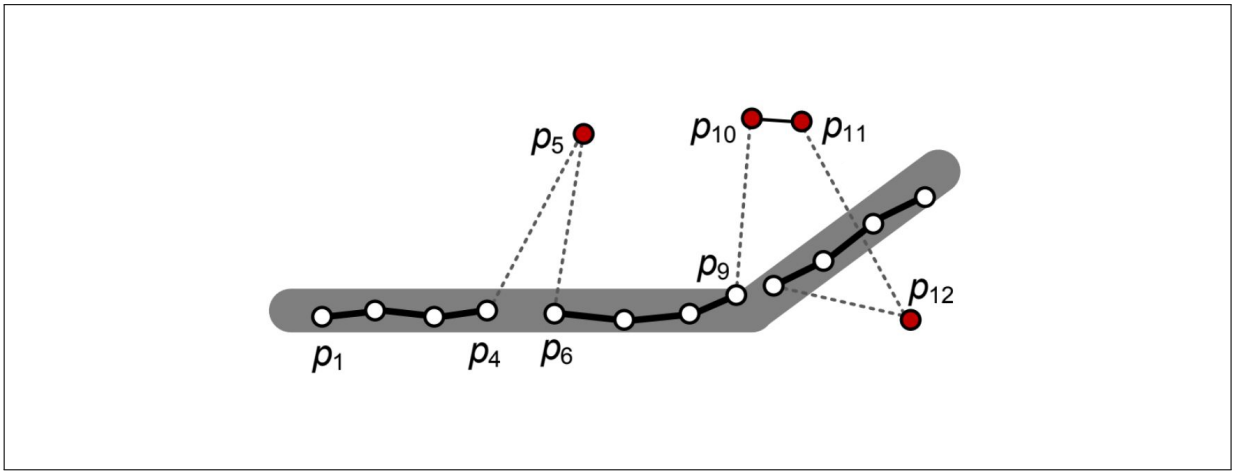
Figura 3 – Trajetórias densas obtidas por meio de dispositivos de *GPS*



Fonte: Elaborado pelo próprio autor (2024).

Ainda com base na Figura 3, é possível notar que a forma de captação ativa dos pontos, consegue empregar maior granularidade aos trajetos, consequentemente, possibilitando uma representação mais fidedigna ao trajeto real. No entanto, taxas amostrais elevadas, falhas em dispositivos eletrônicos, somadas a fatores externos de interferência (como prédios, montanhas e túneis) podem propiciar um aumento na inserção de ruídos nessa classe de dados. Na Figura 4, é apresentado um exemplo básico de ruído onde os pontos (p_5, p_{10}, p_{11} e p_{12}) "fogem" do padrão normal do trajeto. Esse tipo de ruído é conhecido como ruído de valor atípico (do termo em inglês, *outlier*), ocorrendo principalmente quando os dispositivos de captação/geração apresentam falhas de leitura. Esse mesmo tipo de ruído também pode ser observado na trajetória de dados reais da Figura 3, em que a trajetória na cor rosa possui diversos pontos que escapam ("saltam") da rota real do trajeto.

Figura 4 – Pontos com ruído em uma trajetória densa



Fonte: Adaptado de Zheng (2015).

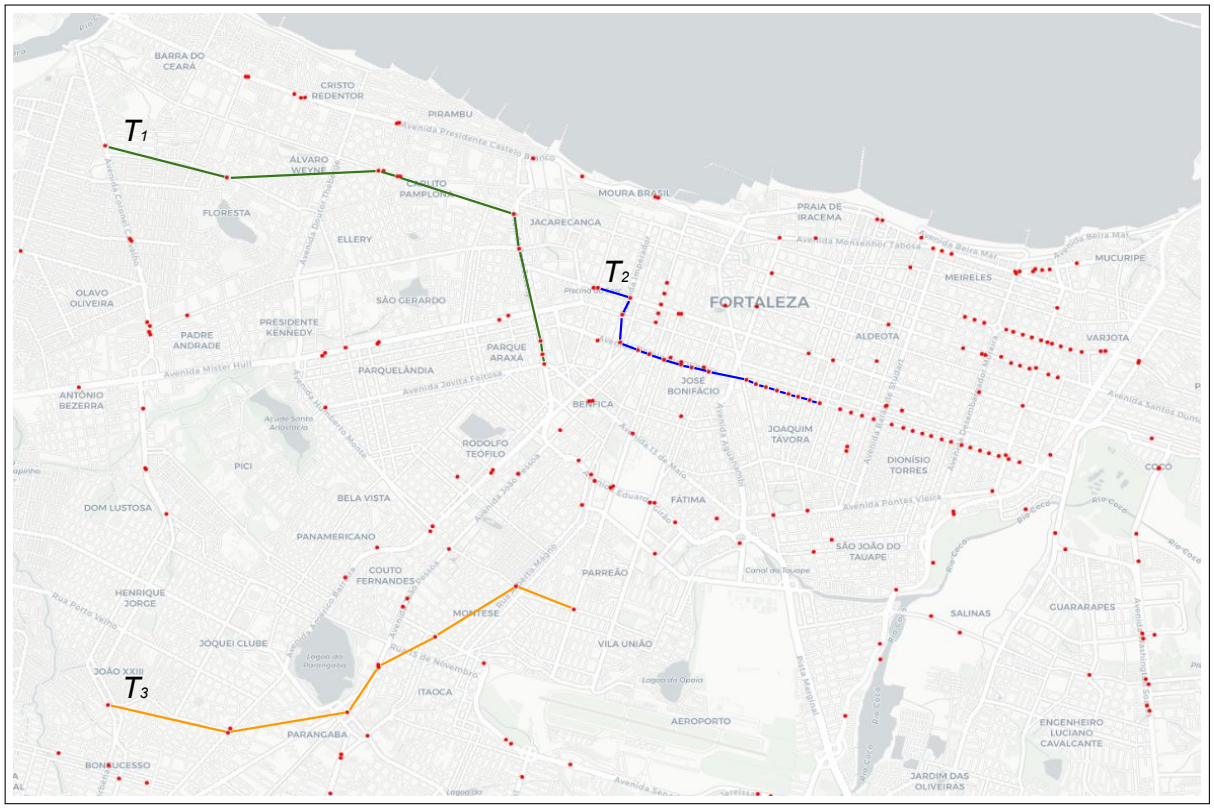
Como visto, trajetórias com alta densidade de pontos são uma categoria de dados bastante desafiadora, tanto pelo gigantesco volume de dados coletados, como pelo fato de estarem mais propícias à inserção de ruídos, necessitando, assim, de um pré-processamento mais cauteloso. No presente trabalho, são consideradas trajetórias densas aquelas com menor distanciamento espacial entre seus pontos e possuem taxas elevadas na amostragem dos mesmos. Assim, no contexto deste trabalho, um conjunto de dados de trajetórias é considerado denso quando a distância média entre os pontos vizinhos for menor ou igual a 20 metros, e a taxa média de gravação/captação dos pontos for menor ou igual a 15 segundos.

2.3 Trajetórias Esparsas

Trajetórias consideradas esparsas são, geralmente, aquelas na qual o processo de geração/captação dos pontos é caracterizado por algum tipo de restrição que implica em um maior distanciamento físico entre seus os pontos. Em Cruz *et al.* (2022), que usam trajetórias de sensores externos de trânsito, *External Sensor Trajectory* (EST), tal restrição é de caráter espacial, pois os dispositivos (sensores de trânsito) responsáveis pela captação de movimento dos objetos móveis (veículos) estão posicionados fisicamente distantes uns dos outros e em uma distribuição não uniforme, produzindo, assim, trajetórias com pontos esparsos entre si.

Devido às restrições das redes de ruas, trajetórias EST possuem taxas amostrais menores e distanciamentos elevados entre seus pontos (sensores) adjacentes. Na rede de ruas usada em Cruz *et al.* (2022), a distância média de um sensor qualquer para seu vizinho mais próximo é de aproximadamente 440 m, número consideravelmente alto quando considerado dados de trajetórias densas, em que na maioria das vezes essa distância média não ultrapassa os 15 m. A fim de elucidar tal esparsidade, a Figura 5 ilustra três exemplos de trajetórias esparsas (T_1, T_2 e T_3) em uma rede de ruas mapeada por sensores trânsito externos e fixos (pontos em vermelho).

Figura 5 – Trajetórias esparsas em uma rede de ruas mapeada por sensores de trânsito



Fonte: Adaptada de Cruz *et al.* (2022).

Ainda com base na Figura 5, é possível observar que devido à distribuição não uniforme dos sensores na rede de ruas, trajetória esparsas possuem características peculiares que as diferenciam bastante uma das outras. Por exemplo, mesmo T_1 tendo quase o dobro do tamanho de T_2, T_1 tem menos pontos (apenas 9 sensores) que T_2 (20 sensores). Esta é apenas uma das características desafiadoras deste domínio de trajetórias, que pode ainda envolver: baixíssimas taxas amostrais, inconsistências de captação, trajetórias incompletas, formato (*shape*) da trajetória muito diferente do trajeto real, dentre outros.

Na literatura, são abordadas também outras formas de trajetórias esparsas, em Damiani *et al.* (2020) as trajetórias usadas são consideradas esparsas, pois são formadas por pontos geoespaciais que indicam eventos dos usuários em uma rede telecomunicação móvel, tais eventos podem ser: início/fim de chamadas, envio/recebimento de mensagens de texto ou *downloads/uploads* de arquivos. Já em Andrade e Gama (2020), as trajetórias são consideradas esparsas porque se baseiam na detecção de pontos de parada (*Stay Points Detection*) e na geração de pontos de interesse *Points of Interest* (POI) para a criação dos seus pontos.

O domínio das trajetórias esparsas apresenta desafios significativos na análise de dados de objetos móveis, dada a natureza inerentemente descontínua dessas trajetórias, exigindo uma abordagem cuidadosa para evitar interpretações incorretas. No presente trabalho, trajetórias são consideradas esparsas com base nas características de distanciamento espacial e temporal entre seus pontos. Assim, no presente trabalho, um conjunto de dados de trajetórias é considerado esparso, se a distância média entre seus pontos vizinhos for maior que 20 m, e a taxa média de gravação/captação dos pontos for maior que 15 segundos.

2.4 Redes Neurais Artificiais

Segundo Haykin (2007), as *Redes Neurais Artificiais* (RNA), comumente denominadas apenas por *Redes Neurais* (RN), tiveram suas pesquisas sempre motivadas pela capacidade do cérebro humano de conseguir processar informações de forma surpreendentemente diferente dos computadores convencionais. O cérebro é um sistema de processamento de informações altamente complexo, não-linear e paralelo que consegue organizar seus componentes estruturais, conhecidos como neurônios, para realizar processamentos complexos (i.e., reconhecimento de padrões, percepção visual, interpretação da fala) com muito mais rapidez que até mesmo os computadores digitais mais sofisticados da atualidade.

Além disso, o cérebro humano está em constante desenvolvimento e aprendizado, as

interações com o ambiente externo possibilitam que novas ligações entre os neurônios sejam estabelecidas instantaneamente, permitindo a adaptação ao ambiente e o surgimento de novos conhecimentos. Assim, os cérebros biológicos possuem um mecanismo de adaptação (função) que é fundamental para que a rede de neurônios do sistema nervoso funcione de maneira adequada. Analogamente, as RNA também precisam de tal função, pois de maneira geral uma rede neural é um sistema/máquina elaborada para "imitar" o cérebro humano na realização de determinadas tarefas.

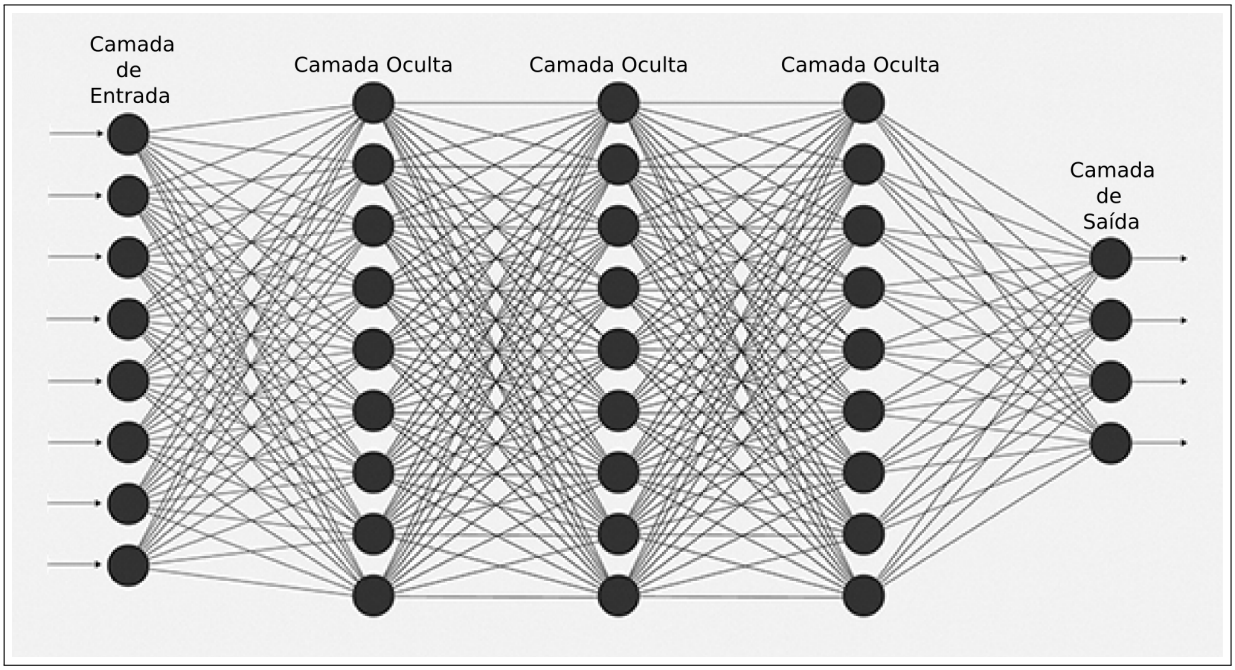
Para que o processo de aprendizagem ocorra eficientemente, assim como as *Redes Neurais Biológicas* (RNB) empregam incontáveis ligações entre seus neurônios, as RNA também estabelecem interligações maciças entre seus neurônios artificias (ou simplesmente "unidades de processamento") a fim de alcançarem bons desempenhos em suas tarefas. Desta forma, as RNA podem ser definidas da seguinte maneira:

> Uma Rede Neural Artificial é um processador maciço paralelamente distribuído e constituído de unidade de processamento simples, com a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso. Ela se assemelha ao cérebro humana em dois aspectos:
> 1. O conhecimento é adquirido pela rede a partir de seu ambiente por meio de um processo de aprendizagem.
> 2. Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido. (Haykin, 2007, p.28)

Quanto ao seu modelo estrutural, as RNA possuem camadas de neurônios por onde as informações passam durante todo o processamento da informação. Como ilustrado na Figura 6, toda a estrutura é composta por camadas verticais de neurônios (pontos pretos), onde cada neurônio de uma camada corrente tem interligação total com os neurônios da camada subsequente. Os tamanhos das camadas (quantidade de neurônios na vertical) e o comprimento da rede (quantidade total de camadas na horizontal) são definidos conforme o propósito da rede, ou seja, conforme a tarefa que ela foi projetada para resolver. A primeira camada, responsável por receber a informação na sua forma inicial, é chamada de *Camada de Entrada*. A última camada, responsável por entregar a informação processada, é a *Camada de Saída*. Já todas as camadas que se encontram entre as camadas de entrada e saída são conhecidas como *Camadas Ocultas* (do termo inglês, *Hidden Layers*) e são elas as responsáveis pelo processamento principal da informação.

As RNA são totalmente compostas por neurônios artificias, tendo suas entradas e saídas representadas por números (ao invés de pulsos elétricos como nas RNB). Cada conexão da rede (linhas que interligam os neurônios) está associada a um peso sináptico (valor numé-

Figura 6 – Estrutura de uma Rede Neural Artificial (RNA)



Fonte: Adaptado de DATA SCIENCE ACADEMY ().

rico previamente definido) e cada camada possui sua função de ativação própria para os seus neurônios.

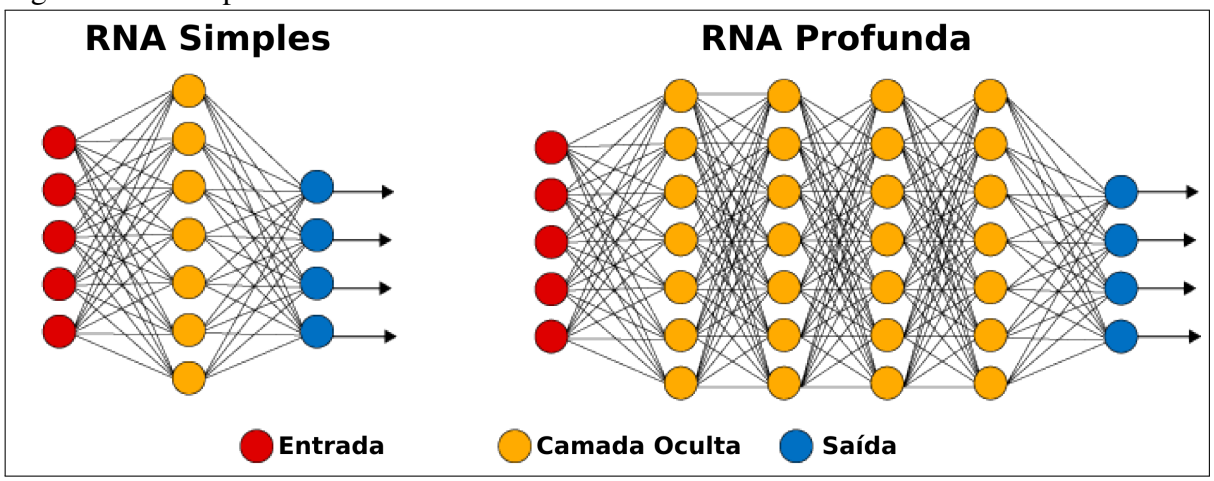
2.5 Redes Neurais Artificiais Profundas

São inúmeros os problemas que podem ser resolvidos utilizando uma rede neural simples com apenas uma ou duas camadas ocultas, obtendo-se resultados satisfatórios. Muitas vezes, uma rede neural simples com apenas uma camada pode modelar funções complexas, bastando para isso que tenha neurônios suficientes (Geron, 2019). Devido a este fato, pesquisadores se convenceram por muito tempo de que não havia necessidade de investir em redes neurais maiores (como muitas camadas ocultas – redes profundas). No entanto, com o passar do tempo, notou-se que redes com um número maior de camadas tinham uma melhor eficiência para problemas de alta complexidade, pois possibilitavam um melhor ajuste fino dos seus hiperparâmetros.

O aumento no número de camadas, mesmo utilizando quantidades exponencialmente menores de neurônios, possibilitou uma execução mais precisa de tarefas ditas de alta complexidade computacional como, por exemplo, o reconhecimento de padrões de voz e imagem. Isso foi possível graças à capacidade das diversas camadas das RNA profundas de segmentar e modelar abstrações de alto nível dos dados envolvidos. A fim de entender a diferença básica entre redes neurais rasas e profundas, a Figura 7 ilustra estes dois modelos de redes neurais, no

lado esquerdo: RNA Simples (rasa) com apenas uma camada oculta, e no lado direito: RNA Profunda com quatro camadas ocultas. Vale ressaltar que, na literatura, uma RNA é considerada profunda quando possui três ou mais camadas ocultas.

Figura 7 – Exemplos de RNA – Rasa e Profunda



Fonte: Adaptado de DATA SCIENCE ACADEMY ().

O uso das RNA profundas tem se tornado presente cada vez mais na vida cotidiana, principalmente na criação de sistemas de inteligência artificial, uma vez que estas redes conseguem modelar tarefas ditas de alta complexidade com muito mais eficiência e granularidade que as redes neurais simples (ou rasas).

2.6 Processamento de Linguagem Natural

Na era moderna da Internet e das mídias sociais, as opiniões, análises e recomendações dos usuários tornaram-se recursos valiosos em diversas áreas, como ciência política e gestão de negócios. Graças ao surgimento de novas áreas de estudo e tecnologias, hoje é possível compreender esses dados de maneira eficiente, semelhante à interpretação humana, por meio de sistemas computacionais (Raschka; Mirjalili, 2019). Segundo Geron (2019), uma dessas áreas emergentes que vem evoluindo bastante é o *Prcessamento de Linguagem Natural* (PLN), muito utilizado em diversas tarefas como a tradução automática, resumo automático, análise sintática e análise de sentimentos. Além dessas, diversas outras aplicações se baseiam, pelo menos em parte, no PLN.

Desta forma, o PLN é entendido como uma subárea da *Inteligência Artificia* (IA), tendo foco nos problemas de geração e compreensão automática de textos que representam a língua humana falada, promovendo o entendimento, manipulação, análise e, até mesmo, geração

instantânea da língua falada. Assim, o PLN desempenha um papel crucial na construção de aplicações modernas, possibilitando processar gigantescos volumes de dados.

2.7 Aprendizado de Representação

O Aprendizado de Representação (do termo inglês, *Representation Learning*) é uma técnica de Aprendizado de Máquina responsável por criar representações numéricas, por meio de *embeddings*, vetores de valores reais, de tamanho fixo e com baixa dimensionalidade (Bengio *et al.*, 2013). Essa técnica permite tratar grandes volumes de dados, possibilitando que as informações intrínsecas dos dados sejam incorporadas nestes vetores. Por exemplo, no caso de dados textuais, os modelos de PLN podem ser treinados a fim de permitir que as informações semânticas do contexto possam ser incorporadas em *embeddings* de espaço vetorial específico R^n. Desta forma, é possível classificar palavras semelhantes como vetores próximos e palavras diferentes como vetores distantes.

Modelos para o aprendizado de representação que tratam palavras e textos são classificados como *Word Embeddings Models*, esses modelos têm como característica fundamental trabalhar com volumosos conjuntos de dados textuais (denominados *corpus* de treino), transformando-os em dados vetoriais com baixa dimensionalidade, que são uma alternativa mais prática e compacta para representar este tipo de informação. Após o treino dos modelos de *word embeddings*, as representações obtidas podem ser aplicadas em diversas tarefas, tais como predição de palavras, classificação textual, completação automática, tradução, entre outras. Segundo Mikolov *et al.* (2013), Devlin *et al.* (2019), devido a estas propriedades e suas vantagens de uso, diversos modelos de *word embeddings* têm sido propostos, destacando-se dentre eles os modelos *Word2Vec* e *Doc2Vec*, ambos projetados pela empresa Google, os quais são utilizados nos experimentos deste trabalho.

Além dos modelos mais simples, como o *Word2Vec* e o *Doc2Vec*, foram testados também modelos mais recentes e avançados conhecidos como *Large Language Models* (LLMs). Esses modelos, como o *BERT (Bidirecional Encoder Representations from Transformers)* e o *SBERT (Sentence BERT)*, demonstraram habilidades de raciocínio semelhantes às humanas. Diferentemente dos modelos de *word embeddings*, que criam representações vetoriais para cada palavra com base nas palavras circunvizinhas, os modelos de LLMs podem gerar vetores para cada palavra (ou segmento de palavras) com base nas características inerentes da estrutura sequencial dos dados (Zhuang *et al.*, 2019). Detalhes sobre o *BERT* e o *SBERT* serão apresentados

nas seções seguintes.

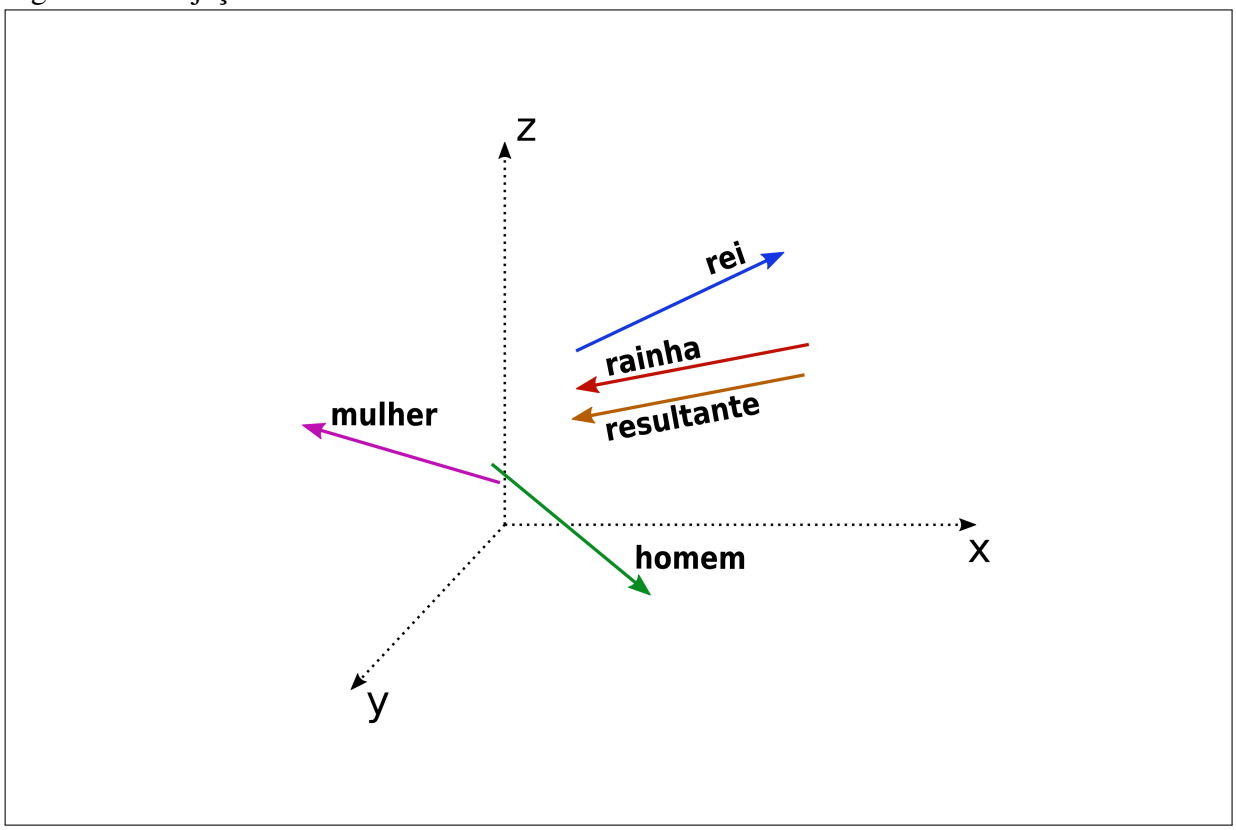
2.8 Modelo Word2Vec

O primeiro modelo de representação adotado neste trabalho foi o *Word2Vec*, um algoritmo de vetorização de palavras (*word embedding*), lançado pela Google (Mikolov *et al.*, 2013), capaz de projetar cada palavra de um conjunto de dados textuais em um espaço vetorial qualquer R^n. Neste modelo, a projeção é realizada de tal forma que as representações vetoriais (*embeddings*) obtidas conseguem incorporar as informações semânticas e/ou sintáticas de um contexto, sendo possível predizer palavras circundantes (ou uma palavra central) dado um determinado contexto de entrada.

Simplificando, o modelo *Word2Vec* é uma rede neural rasa composta apenas por três camadas básicas (Entrada, Projeção e Saída) com o diferencial de usar função de ativação log-linear nos neurônios da sua camada oculta (Projeção). Outro ponto importante é que modelos anteriores (Hinton *et al.*, 1986; Rumelhart *et al.*, 1986) criados para tratar dados textuais eram compostos por diversas camadas ocultas não-lineares, que acabavam deixando estes modelos com uma complexidade computacional muito alta. Diferentemente, o *Word2Vec* é composto apenas por uma única camada oculta, podendo até não conseguir representar os dados com a mesma precisão de um modelo de rede neural mais denso, porém ele é mais vantajoso porque consegue ser treinado com grandes volumes de dados e com muito mais agilidade (menor custo computacional) que os modelos anteriores.

Uma das principais vantagens de se usar o *Word2Vec* está no fato dele conseguir criar representações numéricas (vetores) únicas e eficazes para cada palavra do *corpus* textual utilizado no seu treino, possibilitando, assim, a realização de cálculos matemáticos (que são bem mais rápidos), ao invés de usar a inferência textual que é um processo bastante oneroso e demorado. Para se ter uma noção do uso da ferramenta, a Figura 8 ilustra um exemplo clássico com as quatro palavras: mulher, homem, rei e rainha, com *embeddings* no espaço vetorial R^3, obtidos através do treino do *Word2Vec*. Neste exemplo, as operações matemáticas de soma e subtração vetorial foram aplicadas nos vetores das palavras *rei*, *homem* e *mulher*, obtendo-se um vetor resultante da seguinte forma: $resultante = (rei - homem + mulher)$. É interessante notar que o vetor resultante ficou muito próximo do vetor da palavra rainha, levando a concluir que o modelo foi bem treinado e que conseguiu ter uma boa representação semântica das palavras envolvidas.

Figura 8 – Projeção de vetores treinados com o modelo *Word2Vec*



Fonte: Elaborado pelo autor (2024).

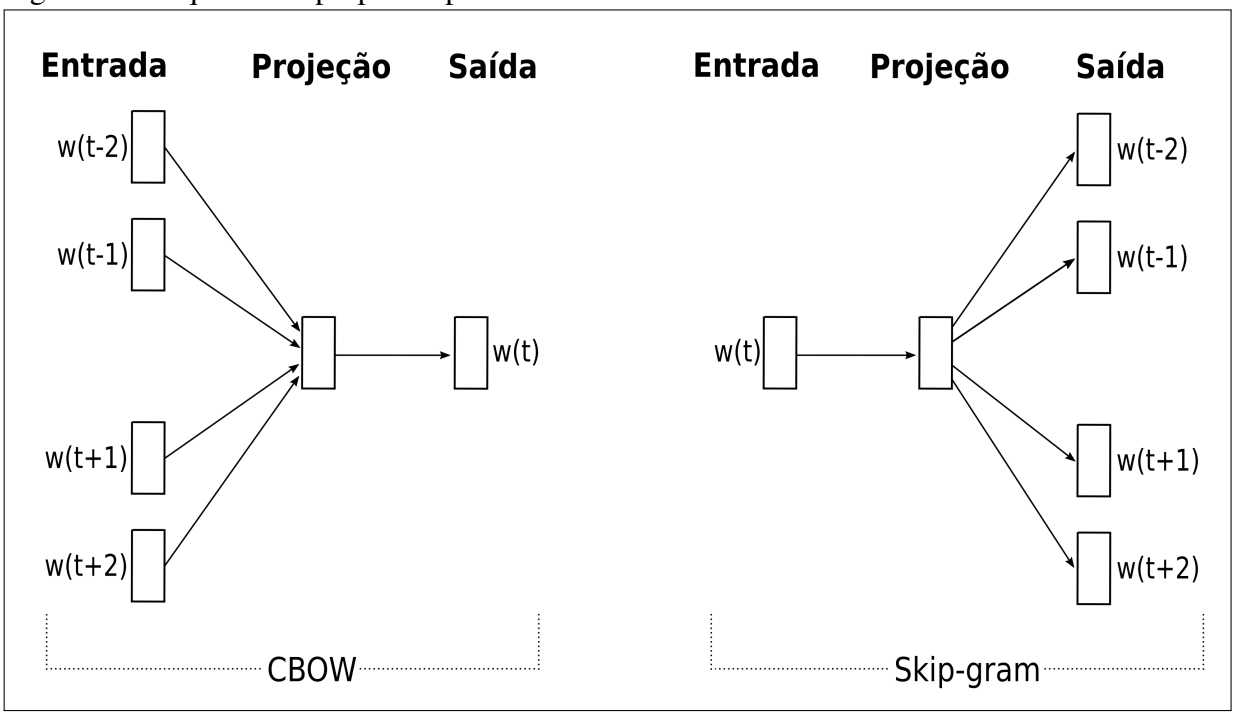
Quanto ao projeto arquitetural da rede neural, o *Word2Vec* possui duas arquiteturas principais que visam aprender representações distribuídas de palavras e minimizar a complexidade computacional observada em modelos de redes neurais elaboradas anteriormente, estes esquemas arquiteturais são: *CBOW* (*Continuous Bag-of-Words*) e *Skip-gram* (*Continuos Skip-gram*), ilustrados com mais detalhes na Figura 9, onde a nomenclatura $w(t)$ referente-se à palavra corrente (ou atual) e suas variações $w(t-n)$ e $w(t+n)$ referem-se às palavras próximas (circunvizinhas) à palavra corrente $w(t)$.

2.8.1 Continuous Bag-of-Words (CBOW)

A primeira arquitetura do modelo tem como base uma rede neural *feedfarward NNLM* (sigla em inglês para *Neural Network Language Model*), onde sua única camada escondida não-linear foi removida para dar lugar a uma camada de projeção (log-linear) compartilhada por todas as palavras do vocabulário. Assim, todas as palavras são projetadas na mesma posição e seus vetores são obtidos por meio do cálculo da média aritmética.

Esta arquitetura é conhecida como "saco de palavras" (*bag-of-words*), pois a ordem histórica das palavras não influencia na projeção. O objetivo da *Continuous Bag of Words*

Figura 9 – Arquiteturas propostas pelo *Word2Vec*



Fonte: Adaptado de Mikolov *et al.* (2013).

(CBOW) é prever uma palavra $w(t)$, dado um determinado contexto circundante de entrada $\{w(t-2), w(t-1), w(t), w(t+1), w(t+2)\}$, percorrendo-se a informação por meio de uma janela deslizante (conjunto de palavras) de tamanho fixo, deslocando-se uma palavra por vez ao longo do texto.

2.8.2 Continuous Skip-gram

Já a segunda arquitetura proposta é similar ao *CBOW*, mas em vez de prever uma palavra baseada em um contexto, ela faz o inverso, prever todo um contexto $\{w(t-2), w(t-1), \cdots, w(t+1), w(t+2)\}$ dada uma determinada palavra $w(t)$ de entrada. Mais precisamente, uma palavra corrente é usada como entrada para um classificador log-linear em uma camada de projeção, assim, um conjunto de palavras (contexto de saída) de um determinado intervalo, antes e depois da palavra corrente, é previsto.

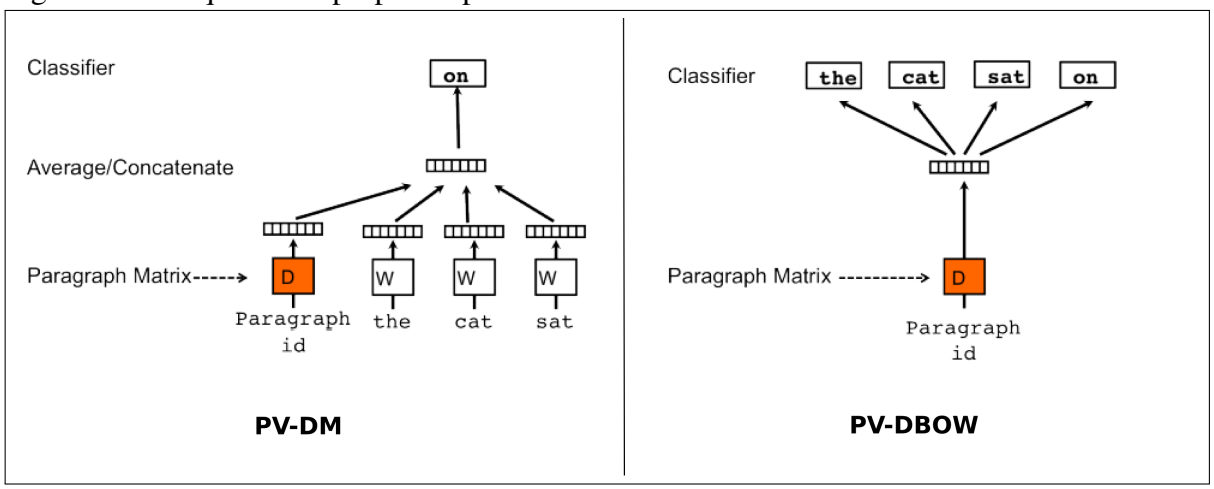
Quanto ao seu treino, a arquitetura *Skip-gram* utiliza pares n-gram (palavra_alvo, contexto_da_palavra_alvo) buscando maximizar a probabilidade de predizer um contexto de palavras a partir de uma palavra central de entrada. Para este cálculo probabilístico, a camada de projeção recebe o vetor da palavra corrente e projeta a viabilidade de cada palavra do contexto de saída.

2.9 Modelo Doc2Vec

Ao contrário do *Word2Vec*, o modelo *Doc2Vec* é usado para criar representações vetoriais (*embeddings*) para sequências inteiras de palavras (frases, parágrafos ou documentos), produzindo representações vetoriais únicas para estes grupos textuais, permitindo que textos parecidos obtenham vetores semelhantes entre si. Este modelo é amplamente utilizado e independe do tamanho dos textos/documentos utilizados em seus dados de treinamento. Desta forma, enquanto o *Word2Vec* projeta vetores de dimensões fixas para cada palavra, o modelo *Doc2Vec* fornece vetores com as mesmas características para cada unidade textual (frase, parágrafo ou documento) presente no seu *corpus* de treinamento.

Em Le e Mikolov (2014) o modelo *Doc2Vec* é definido com base no *Word2Vec*, tendo como principal diferença a adição da informação do identificador de parágrafos (*paragraph ID*) à entrada do modelo, como ilustrado na Figura 10. Semelhante ao *Word2Vec*, o *Doc2Vec* conta também com dois modelo arquiteturais, os quais são: *Distributed Memory version of Paragraph Vector* (PV-DM), análogo ao *CBOW*, e o *Distributed Bag of Words version of Paragraph Vector* (PV-DBOW), análogo ao *Skip-gram*.

Figura 10 – Arquiteturas propostas pelo *Doc2Vec*



Fonte: Adaptado de Le e Mikolov (2014).

Dessa forma, o objetivo do *Doc2Vec* é gerar representações numéricas únicas para documentos textuais, independentemente de seus tamanhos. Para isso, Le e Mikolov (2014) usaram o *Word2Vec* como base para obter as representações vetoriais de todas as palavras envolvidas no treinamento e, simplesmente, adicionaram um identificador único de documentos (*paragraph ID*) que também passou a ser treinado e vetorizado junto as demais palavras. Na arquitetura *PV-DM* o modelo atua como uma memória que lembra o que está faltando em um

contexto corrente, que pode ser uma palavra, tópico de um parágrafo ou título de um documento completo. Já na arquitetura *PV-DBOW*, o modelo age de maneira inversa, um identificador de parágrafo é recebido na sua camada de entrada, vetorizado na camada escondida do modelo que, por sua vez, usa este vetor na camada de saída para predizer o contexto do documento de entrada.

2.10 Modelo BERT

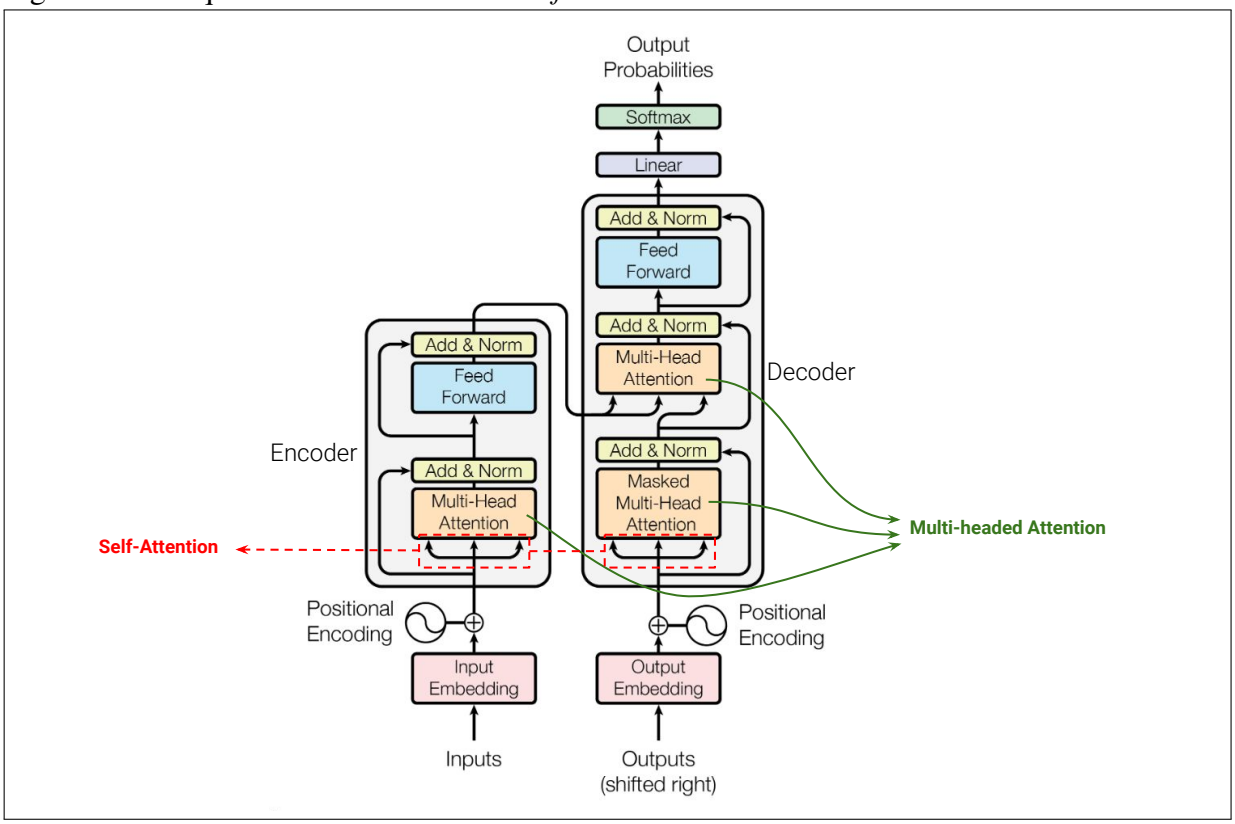
O modelo *Bidirectional Encoder Representations from Transformers* (BERT), proposto em Devlin *et al.* (2019), é um modelo de aprendizado de máquina para Processamento de Linguagem Natural (PLN) baseado na arquitetura de rede neural *transformers*, basicamente o modelo *BERT* pode ser entendido como uma pilha de codificadores *transformers*, possuindo 12 codificadores em sua versão base e 24 na versão estendida. De acordo com Vaswani *et al.* (2017), os codificadores (*transformers*) são modelos baseados em mecanismos de atenção, dispensando completamente as recorrências e convoluções usadas em modelos de redes neurais anteriores.

As redes neurais do tipo *transformers* conseguem aprender o valor semântico de um contexto através do monitoramento dos dados sequenciais da informação textual, mapeando as relações existentes entre cada palavra nas frases. Para isso, ao invés de usar a recorrência, o modelo é totalmente voltado aos mecanismos de atenção (ou autoatenção) para mapear as dependências globais entre todas as entradas e saídas do modelo.

Semelhante às redes neurais convencionais, modelos *transformes* possuem empilhamento de blocos, onde esses blocos são conjuntos de elementos codificadores/decodificadores que realizam todo o processamento dos dados. Estruturalmente, a arquitetura é bem semelhante às redes anteriores, porém, nas redes *transformers*, foram incluídas estrategicamente pequenas modificações que são fundamentais nesta arquitetura. A primeira delas, como ilustra a Figura 11, foi a adição das funções de atenção (*Multi-headed Attention e Self-Attention*). A segunda, foi a inclusão de codificadores de posição (*Positional Enconding*) para marcar as entradas e saídas da rede eficientemente, realizando cálculos matemáticos para identificar como cada elemento de entrada/saída se relacionam.

Muitas abordagens na literatura (Mikolov *et al.*, 2013; Peters *et al.*, 2018; Radford *et al.*, 2018) utilizam modelos *transformers* (ou modelos de PLN anteriores) para resolver alguma tarefa específica no campo do Processamento da Linguagem Natural. Tais abordagens compartilham o mesmo objetivo, a representação da linguagem escrita, e para isto usam modelos de aprendizado de máquina que fazem a análise/mapeamento das sentenças (frases) apenas em

Figura 11 – Arquitetura de uma rede *Transformers*.



Fonte: Adaptado de Vaswani *et al.* (2017)

um sentido (da direita à esquerda ou o inverso), modelos estes conhecidos como modelos de linguagem unidirecional. Segundo Devlin *et al.* (2019), a maior limitação desses modelos se dá pelo fato deles usarem a arquitetura padrão de análise unidirecional. Por exemplo, em um modelo *transformers* que usa a arquitetura "esquerda à direita", cada *token* pode atender apenas aos *tokens* anteriores nas suas camadas de autoatenção. Essas restrições impactam negativamente nas tarefas ao nível de sentenças (frases) e podem ser muito prejudiciais na aplicação de abordagens que necessitam de um ajuste fino (*finetuning*) ao nível de *tokens*, como na tarefa de perguntas e respostas, onde é crucial incorporar/entender o contexto em ambas as direções.

A fim de solucionar essas restrições foi que o modelo de arquitetura bidirecional *BERT* foi proposto por Devlin *et al.* (2019). Os autores enfatizam que o modelo *BERT* consegue "aliviar" as restrições dos modelos de linguagem unidirecional usando a abordagem de treino "Modelo de Linguagem Mascarada" (do termo inglês, *Masked Language Model - MLM*). Esta abordagem faz o mascaramento randômico de um pequeno percentual de *tokens* de entrada, e tenta predizer o *id* do vocabulário original da palavra que foi mascarada, tomando por base apenas seu contexto. Ao contrário dos modelos de linguagem unidirecional, a abordagem *MLM* permite que as representações façam a junção do contexto direito e esquerdo das sentenças,

possibilitando o treinamento de redes *transformers* bidirecionais profundas. Além da abordagem de modelo de linguagem mascarada (*MLM*), a arquitetura *BERT* também permite o uso da abordagem "Predição da Próxima Sentença" (do termo inglês, *Next Sentence Prediction - NSP*) que consiste em alimentar o modelo com pares de sentenças para prever se tais sentenças estão relacionadas entre si.

Outro ponto crucial é que o modelo *BERT* passa por duas etapas principais: o Pré-treino (*Pre-training*) e o Ajuste Fino (*Fine-Tuning*). Durante o pré-treino, o modelo é treinado com os dados não rotulados em diferentes tarefas. Já no ajuste fino, o modelo é inicializado com os mesmos parâmetros do pré-treino, porém, logo em seguida, todos os parâmetros são ajustados usando dados rotulados para uma tarefa específica. Assim, a grande vantagem do *BERT* é que diversos modelos pré-treinados em imensas bases de dados textuais podem ser facilmente carregados e utilizados em alguma tarefa específica, fazendo apenas o ajuste fino dos seus hiperparâmetros para tal tarefa.

2.11 Modelo SBERT

O modelo *SBERT*, *Sentence BERT* (Reimers; Gurevych, 2019), é uma variação do modelo *BERT* projetada para gerar incorporações (*embeddings*) de sentenças. Uma vez que o modelo *BERT* foi criado para entender contextos bidirecionais de palavras através do seu treino, o *SBERT* leva essa mesma abordagem para o nível de sentenças. O diferencial do *SBERT* é que ele utiliza uma estrutura siamesa de treino, onde as sentenças semelhantes são mapeadas para espaços vetoriais próximos, enquanto sentenças diferentes são mapeadas para espaços vetoriais mais distantes.

A forma de treinamento do *SBERT* é conhecida como *Contrastive Learning*, a qual se concentra em aprender representações semelhantes para sentenças semanticamente relacionadas. O foco do treinamento consiste em minimizar a distância entre *embeddings* de sentenças semelhantes e maximizar a distância entre *embeddings* de sentenças diferentes.

Além disso, o *SBERT* recorre a modelos *BERT* pré-treinados em grandes conjuntos de dados, em seguida realiza o ajuste fino para uma tarefa específica, capturando informações contextuais e semânticas de cada sentença. Os *embeddings* pré-treinados, então, podem ser usados em uma variedade de tarefas, como clusterização de sentenças, recuperação de informações e similaridade entre sentenças. Desta forma, o *SBERT* pode oferecer apoio significativo para comparações tradicionais de *embeddings* de palavras ou frases.

2.12 Métodos Clássicos para o Cálculo de Similaridade entre Trajetórias

O cálculo da similaridade entre trajetórias é peça fundamental na análise de dados geoespaciais, dada sua aplicabilidade na resolução de diversos problemas de mobilidade. A seguir, são apresentadas algumas das técnicas padrão para determinar o grau de similaridade entre trajetórias. Essas técnicas foram implementadas no presente trabalho com o intuito de fornecer uma comparação adicional aos modelos de linguagem.

2.12.1 Distância euclidiana

Quando se têm sequências discretas de tamanhos iguais em um espaço vetorial R^n, a técnica comumente utilizada para medir a distância entre tais objetos é a *distância euclidiana*, técnica relativamente simples e intuitiva, o que a tornou uma das técnicas mais bem adotada na literatura. Além da sua simplicidade, a distância euclidiana tem diversas outras vantagens, uma delas é que sua complexidade é de caráter linear, possibilitando uma fácil implementação. A distância euclidiana foi proposta como uma medida de similaridade entre duas series temporais, sendo considerada uma das funções de distância mais utilizadas desde os anos 60 (Faloutsos *et al.*, 1994; Keogh; Pazzani, 2000).

Como trajetórias também são consideradas series temporais, a distância euclidiana pode ser facilmente adotada para calcular a distância entre elas (Li *et al.*, 2018a; Sanderson; Wong, 1980). Assim, para duas trajetórias T_1 e T_2 que tenham o mesmo comprimento n, a distância euclidiana $d_E(T_1, T_2)$ é definida pela Equação 2.1.

$$d_E(T_1, T_2) = \frac{\sum_{i=1}^{n} d(p_{1,i}, p_{2,i})}{n} \tag{2.1}$$

Onde $p_{1,i}$ e $p_{2,i}$ é o i-ésimo par de pontos entre T_1 e T_2, sendo d a distância real entre esses dois pontos. Desta forma, a distância euclidiana tem complexidade de tempo $O(n)$, tratando-se de uma implementação relativamente simples. Porém, o seu principal problema é que ela requer que as trajetórias comparadas sejam exatamente do mesmo tamanho, o que nem sempre é possível nas bases de dados de trajetórias, acabando por inviabilizar a comparação ponto a ponto entre os pares de trajetórias de tamanhos diferentes.

2.12.2 Distância DTW – Dynamic Time Warping

A *Dynamic Time Warping* (DTW) é uma das técnicas mais utilizadas quando se precisa determinar a similaridade entre duas séries temporais. O algoritmo *DTW* visa encontrar o alinhamento global ótimo entre estas duas séries, explorando suas distorções temporais, criando um mapeamento dinâmico entre elas. Este mapeamento permite determinar um valor numérico de distância que, a partir dele, possibilita identificar um grau de similaridade entre as duas sequências analisadas (Salvador; Chan, 2007).

Segundo Shuncheng *et al.* (2019), a técnica *DTW* existe há mais de um século e foi inicialmente introduzida para o cálculo da distância entre séries temporais. No entanto, foi somente na década de 80 que surgiram os primeiros trabalhos (Kruskal, 1983; Roucos; Dunham, 1987) que introduziram a *DTW* para o cálculo da medida de distância/similaridade entre trajetórias geoespaciais. Desde então, a distância *DTW* se tornou uma das técnicas de distância para trajetórias mais utilizada na literatura.

A alta popularidade da distância *DTW* está diretamente relacionada ao fato dela ser facilmente implementada recorrendo-se à programação dinâmica (ou recursiva) e por sua aplicabilidade em diversas áreas, como: reconhecimento de voz, reconhecimento de assinatura, processamento de sinais, classificação de padrões, alinhamento musical, dentre outras. Assim, a técnica *DTW* pode ser usada em qualquer tipo de dado que obedeça uma certa ordem temporal.

No caso deste trabalho, a distância *DTW* é usada para o cálculo de similaridade entre trajetórias de objetos móveis, computando-se as distâncias entre todos os pares possíveis de trajetórias do conjunto de dados de teste. Uma vez que a *DTW* faz uma comparação (*match*) entre todos os pontos de duas sequências temporais, é possível, assim, identificar o caminho de menor custo (*warping path*) entre duas trajetórias quaisquer, ou seja, o caminho com melhor alinhamento entre as duas trajetórias comparadas. Mais especificamente, dadas duas trajetórias T_1 e T_2 de comprimentos (n e m), a distância *DTW* é definida recursivamente pela Equação 2.2.

$$d_{DTW}(T_1, T_2) = \begin{cases} 0, & \text{se } n = 0 \text{ e } m = 0 \\ \infty, & \text{se } n = 0 \text{ ou } m = 0 \\ d(\text{Topo}(T_1), Topo(T_2)) + \min\{d_{DTW}(T_1, \text{Resto}(T_2)), & \\ d_{DTW}(\text{Resto}(T_1), T_2), & \\ d_{DTW}(\text{Resto}(T_1), \text{Resto}(T_2))\} & \text{caso contrário} \end{cases} \tag{2.2}$$

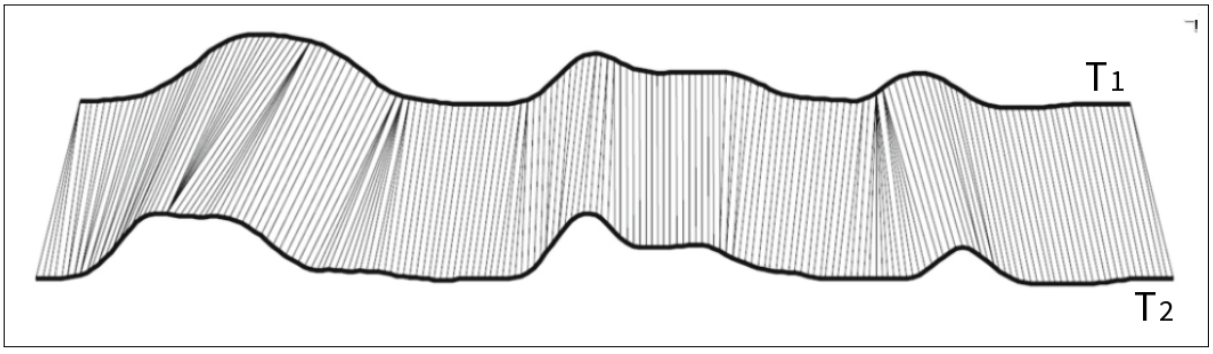
Na Equação 2.2, T_1 e T_2 representam os pontos geoespaciais (lon, lat) das duas trajetórias analisadas, n e m são seus respectivos comprimentos, $Topo()$ e $Resto()$ são os conjuntos de pontos definidos na seção 2.1 e d é a distância espacial (harvesine), Equação 2.3, usada para o cálculo de distância entre dois pontos quaisquer do globo terrestre a partir de suas informações de latitudes (φ) e longitudes (λ).

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi 2 - \varphi 1}{2}\right) + \cos\varphi 1 . \cos\varphi 2 . \sin^2\left(\frac{\lambda 2 - \lambda 1}{2}\right)}\right) \tag{2.3}$$

Quanto à execução do algoritmo, a distância *DTW* tem complexidade $O(nm)$, pois é necessário fazer a comparação entre todos os pontos das duas trajetórias. Precisa-se, então, identificar a distância de cada ponto p_i da trajetória T_1 para os demais pontos p_j da trajetória T_2 a fim de encontrar os pares (p_i, p_j) de menores distâncias que serão usados na construção do caminho de custo mínimo (*warping path*) para o cálculo final da distância *DTW*. Os Algoritmos 1 e 2 apresentam, por meio da programação dinâmica, os passos utilizados neste trabalho para efetuar o cálculo da distância *DTW* entre duas trajetórias geoespaciais quaisquer T_1 e T_2.

Como o algoritmo *DTW* precisa percorrer todos os pontos das trajetórias T_1 e T_2 de comprimentos n e m, o cálculo da *DTW* é considerado de ordem quadrática $O(n^2)$, tornando-o inviável de ser utilizado em volumosos conjuntos de dados. Para um melhor entendimento, a Figura 12 ilustra a sequência de pares de pontos, com menores distâncias, criados no alinhamento de menor custo do cálculo da *DTW* para duas trajetórias T_1 e T_2.

Figura 12 – Combinação (*match*) de pares de pontos entre duas trajetórias usando a *DTW*



Fonte: Adaptado de Shuncheng *et al.* (2019).

Algoritmo 1: Cálculo da distância *DTW*

Entrada: trajetórias geoespaciais T_1 e T_2.
Saída : valor da distância *DTW* entre T_1 e T_2.

```
1  início
2      D ← [n,m] ;                                   /* Matriz de zeros n x m */
3      distDTW ← 0;
4      para j := 0 até m-1 faça
5          D[0, j] ← d(p_0, p_j) ;                   /* Preenche a primeira linha de D */
6      fim
7      para i := 0 até n-1 faça
8          D[0, j] ← d(p_0, p_i) ;                   /* Preenche a primeira coluna de D */
9      fim
       /* Preechendo o restante da matriz distâncias entre T_1 e T_2     */
10     para i := 1 até n-1 faça
11         para j := 1 até m-1 faça
12             custo ← d(p_i, p_j);
13             D[i, j] ← custo + min(D[i−1, j],        // insertion
14                                   D[i, j−1],        // deletion
15                                   D[i−1, j−1])      // match
16         fim
17     fim
18     wp ← warping_path(D) ;                        /* Lista de comprimento |wp| */
19     para i := 0 até |wp|-1 faça
20         distDTW ← distDTW + D[wp[i][0], wp[i][1]];
21     fim
22     distDTW ← distDTW/|wp|;
23     retorne distDTW
24 fim
```

2.12.3 Distância EDR – Edit Distance on Real Sequence

De acordo com Levenshtein (1965), a técnica *Edit Distance* (ED) foi usada pela primeira vez em 1966 para medir a similaridade entre duas sequências textuais (*strings*). A distância *Edit Distance on Real Sequence* (EDR) baseia-se no método da *ED*, alinhando duas series temporais na busca da contagem mínima de (*scores*) de edição (inserção, exclusão e/ou substituição) necessários para transformar uma sequência na outra. Esta técnica, por ser considerada robusta e de fácil implementação, pode ser aplicada em diversas áreas que tem como princípio o tratamento de sequências, como o processamento de linguagem, bioinformática, reconhecimento de padrões, dentre outros.

Como trajetórias são sequências espaciais semelhantes às sequências textuais, a similaridade espacial entre trajetórias pode ser facilmente calculada usando a distância *EDR*.

Algoritmo 2: Cálculo do caminho de menor custo (*Warping Path*)

Entrada: matriz D de distâncias entre as trajetórias T_1 e T_2.
Saída : lista com os índices do *Warping Path* em D.

```
1  início
2      n ← Quantidade de linhas em D;
3      m ← Quantidade de colunas em D;
4      P ← [ ] ;                                          /* Lista vazia */
5      enquanto n > 0 ou m > 0 faça
6          se n = 0 faça
7              valor_min ← (0, m − 1);
8          fim
9          senão, se m = 0 faça
10             valor_min ← (n − 1, 0);
11         fim
12         senão, faça
13             valor ← min(D[n − 1, m − 1], D[n − 1, m], D[n, m − 1]);
14             se valor = D[n − 1, m − 1] faça
15                 valor_min ← (n − 1, m − 1);
16             fim
17             senão, se valor = D[n − 1, m] faça
18                 valor_min ← (n − 1, m);
19             fim
20             senão, faça
21                 valor_min ← (n, m − 1);
22             fim
23         fim
24         P ← P + valor_min ;                    /* P.append(valor_min) */
25     fim
26     retorne P
27 fim
```

Shuncheng *et al.* (2019) definem que a distância *EDR* (d_{EDR}) entre duas trajetórias quaisquer T_1 e T_2 de comprimentos n e m, respectivamente, pode ser calculada pelo número de edições (inserções, exclusões e/ou substituições) necessárias para transformar T_1 em T_2.

Quando são usados dados de trajetórias provenientes de dispositivos de *GPS*, o cálculo da distância *EDR* requer a adição de um valor de subcusto, uma vez que é muito difícil que os pontos de T_1 e T_2 tenham as mesmas coordenadas geoespaciais (necessárias para identificar a equivalência entre pontos – *match points*). Então, para calcular a distância entre duas trajetórias T_1 e T_2, o cálculo da distância *EDR* considera que os pontos $p_i(p_i \in T_1)$ e $p_j(p_j \in T_2)$ são equivalentes se a distância d (*haversine* ou a distância euclidiana) entre p_i e p_j estiver dentro do intervalo ε de subcusto. Dessa forma, são usadas duas equações para o cálculo da distância *EDR* de forma recursiva: a Equação 2.4 para encontrar o valor do subcusto e a Equação 2.5 para

o cálculo específico da distância *EDR* entre T_1 e T_2.

$$subcusto(p_i, p_j) = \begin{cases} 0, & d(p_i, p_j) \leq \varepsilon \\ 1, & \textit{caso contrário} \end{cases} \tag{2.4}$$

$$d_{EDR}(T_1, T_2) = \begin{cases} n, & \textit{se } m = 0 \\ m, & \textit{se } n = 0 \\ min\{d_{EDR}(Resto(T_1), Resto(T_2)) + subcusto(Topo(T_1), Topo(T_2)), & \\ d_{EDR}(Resto(T_1), T_2) + 1, & \\ d_{EDR}(T_1, Resto(T_2)) + 1\} & \textit{caso contrário} \end{cases} \tag{2.5}$$

A seguir, é descrito em pseudo código (Algoritmo 3) os passos necessários utilizados para implementar a Equação 2.5.

Algoritmo 3: Cálculo da distância *EDR*

Entrada: trajetórias T_1 e T_2 no formato sequencial de células.
Saída : valor da distância *EDR* entre T_1 e T_2.

```
1  início
2      n ← |T1| ;                                  /* Comprimento de T1 */
3      m ← |T2| ;                                  /* Comprimento de T2 */
4      se n = 0 faça
5          retorne m;
6      fim
7      senão, se m = 0 faça
8          retorne n;
9      fim
10     senão, faça
11         retorne min(EDR(Resto(T1), Resto(T2)) +
               subcusto(Topo(T1), Topo(T2)), EDR(Resto(T1), T2) +
               1, EDR(T1, Resto(T2)) + 1));
12     fim
13 fim
```

2.12.4 Distância LCSS - Longest Common Subsequence

O valor de similaridade entre duas séries textuais quaisquer S_1 e S_2 (*strings*) pode ser entendido como o comprimento da maior subsequência comum entre estas duas séries (Shuncheng *et al.*, 2019). Como trajetórias geoespaciais são conjuntos sequenciais de pontos, semelhante às sequências textuais, o valor da maior subsequência em comum, *Longest Common Subsequence* (LCSS), pode ser usado para identificar a similaridade entre duas trajetórias quaisquer T_1 e T_2, tendo como resultado o comprimento da maior subsequência de pontos entre T_1 e T_2 (Vlachos *et al.*, 2002).

Algoritmo 4: Cálculo da distância *LCSS*

Entrada: Trajetórias T_1 e T_2 no formato sequencial de células.
Saída : Valor da distância *LCSS* entre T_1 e T_2.

```
1  início
2      n ← |T1| ;                          /* Comprimento de T1 */
3      m ← |T2| ;                          /* Comprimento de T2 */
4      se n = 0 ou m = 0 faça
5          retorne 0;
6      fim
7      senão, se equivalencia(Topo(T1), Topo(T2)) = 1 faça
8          retorne d_LCSS(Resto(T1), Resto(T2)) + 1;
9      fim
10     senão, faça
11         retorne max{d_LCSS(T1, Resto(T2)), d_LCSS(Resto(T1), T2)};
12     fim
13 fim
```

No entanto, assim como na distância *EDR*, é muito difícil que os pontos geoespaciais $p_i(p_i \in T_1)$ e $p_j(p_j \in T_2)$ sejam exatamente iguais, ou seja, caiam na mesma geolocalização (lon, lat). Isso ocorre na maioria das vezes devido às características inerentes dos dispositivos de *GPS*, do trajeto, das taxas de amostragem ou, até mesmo, das interferências causadas por ruídos. Desta forma, um limite referencial ε de distância também deve ser adotado para indicar a equivalência (*match*) entre os pontos das duas trajetórias analisadas. A Equação 2.6 demostra o processo recursivo para o cálculo do valor *LCSS* entre T_1 e T_2 com comprimentos m e n, respectivamente:

$$LCSS(T_1,T_2) = \begin{cases} 0, & se\ m = 0\ ou\ n = 0 \\ d_{LCSS}(Resto(T_1), Resto(T_2)) + 1, & se\ d(Topo(T_1), Topo(T_2)) \leq \varepsilon \\ max\{d_{LCSS}(T_1, Resto(T_2)),\ d_{LCSS}(Resto(T_1), T_2)\}, & caso\ contrário \end{cases} \quad (2.6)$$

$$d_{LCSS}(T_1, T_2) = Tamanho(T_1) + Tamanho(T_2) - 2 \cdot LCSS(T_1, T_2). \quad (2.7)$$

Desta forma, o valor de similaridade *LCSS* calcula o comprimento da maior subsequência equivalente entre T_1 e T_2, usando a função de subsusto da Equação 2.4 para encontrar o *match* de pontos e, assim, conseguir definir as subsequências de maior comprimento. E tomando como base a medida de similaridade da Equação 2.6, o cálculo da distância *LCSS* entre duas trajetórias T_1 e T_2 é dado pela Equação 2.7. Na sequência, o Algoritmo 4 ilustra o conjunto de passos utilizados na implementação do cálculo do valor de similaridade *LCSS* de maneira recursiva.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados que exploram o uso de *embeddings*, na representação de trajetórias, obtidos via treinamento de modelos de aprendizado profundo. A maioria desses trabalhos visa solucionar o problema do cálculo da similaridade espacial entre trajetórias por meio de representações vetoriais (*embeddings*) obtidas com o auxílio da discretização por uma grade espacial uniforme. A menor parte dos trabalhos usa abordagens de discretização diferentes (e.g., Pontos de Interesse ou Rede de Segmentos), focando no cálculo da similaridade ou outra tarefa específica. Ao final, o Quadro 1 contendo as comparações entre todos os trabalhos relacionados e este é apresentado com base nas características específicas de cada trabalho.

3.1 Deep Representation Learning for Trajectory Similarity Computation

Trajetórias de objetos móveis geram grandes volumes de dados, considerados altamente complexos em termos de processamento, desde a captura até o armazenamento e a recuperação. Além disso, interpretar estes dados eficientemente, de modo a recuperar/tratar a informação corretamente, é uma tarefa desafiadora, principalmente quando esses dados são coletados com baixa taxa amostral, pouca uniformidade e propícios à inserção de ruídos.

Com base neste cenário e no desafio de se determinar a similaridade espacial entre trajetórias, Li *et al.* (2018b) propuseram uma nova abordagem, chamada *t2vec* (do termo em inglês, *trajectory to vector*), capaz de representar trajetórias em um novo domínio (espaço vetorial) e, ainda, promover o cálculo de similaridade espacial entre trajetórias de forma bastante eficaz. O modelo *t2vec* é um *autoencoder* que utiliza diversas camadas *GRU* (*Gated Recurrent Unit*) para codificar trajetórias discretizadas em *embeddings*. Assim, o *t2vec* é treinado para reconstruir as trajetórias discretizadas a partir de versões com menor taxa amostral (*downsampled*) e versões adicionadas de ruídos (*distorting*).

As representações (*embeddings*) aprendidas pelo *t2vec* foram projetadas para serem robustas a dados com baixa taxa amostral, não uniformes e com ruídos, auxiliando no cálculo da similaridade espacial entre trajetórias dessa natureza. As representações vetoriais obtidas por meio do aprendizado alcançaram um tempo de execução linear, $O(n+|v|)$, (onde n é o tamanho da trajetória, $|v|$ é o comprimento do vetor) para cálculo de similaridade espacial entre duas trajetórias. Enquanto que, maioria das técnicas padrão, que lidam com os dados em sua forma

bruta (long, lat), geralmente alcançam uma complexidade quadrática, $O(n^2)$, para essa tarefa.

Embora trabalhos mais recentes tenham superado o *t2vec* em termos de qualidade dos *embeddings* produzidos (como alguns dos trabalhos relacionados abaixo), o *t2vec* foi considerado referência neste estudo, pois seu código: está disponível publicamente, permite resultados reproduzíveis, e sua qualidade e a aplicabilidade foram minuciosamente avaliadas em pesquisas recentes (Taghizadeh *et al.*, 2021), ressaltando sua relevância como abordagem de referência.

3.2 Computing Trajectory Similarity in Linear Time: A Generic Seed-guided Neural Metric Learning Approach

Alguns modelos baseados em aprendizado profundo que vieram após o *t2vec* adotaram uma abordagem diferente, ao invés de aprender *embeddings* de trajetórias por meio de uma tarefa de reconstrução, eles foram treinados para aproximar funções de similaridade bem conhecidas e, ao mesmo tempo, garantir o cálculo da similaridade entre trajetórias em tempo linear $O(n)$.

Um modelo pertencente a esta classe de abordagem é o *NeuTraj* (Yao *et al.*, 2019). Em sua essência, o modelo inicialmente calcula uma matriz de distâncias D aplicando alguma medida de similaridade alvo entre "trajetórias sementes", i.e., uma pequena fração de trajetórias extraídas de forma aleatória do conjunto de dados original (*dataset*). Esta matriz é então normalizada em uma matriz de similaridade S, preparando, então, o cenário para o processo de aprendizagem.

Além disso, um novo componente, crucial na arquitetura do *NeuTraj*, foi elaborado: o módulo *SAM (Spatial Attention Memory)*, cujo objetivo é capturar características e correlações espaciais importantes, atualizando dinamicamente um tensor de memória com informações espaciais das trajetórias. Este módulo é então treinado de acordo com um procedimento de amostragem ponderada por distância e um objetivo de perda de classificação, que garantem um treinamento eficiente do modelo, priorizando as trajetórias sementes que estão mais próximas no espaço de incorporação. Essa ênfase seletiva permite que o *Neutraj* aprenda uma função que mapeia trajetórias em um espaço de baixa dimensão, onde as relações geométricas refletem de perto aquelas definidas pela medida de similaridade alvo.

3.3 T3S: Effective Representation Learning for Trajectory Similarity Computation

O avanço no uso de novas tecnologias capazes de gerar dados de trajetórias, como sensores e celulares, impulsionou a proliferação desses dados em um amplo espectro de aplicações. Os autores em Yang *et al.* (2021) relatam que dados de trajetórias geoespaciais são cada vez mais cruciais para aplicações do cenário atual e a maioria delas precisa lidar com um problema fundamental deste domínio de dados: a avaliação de similaridade espacial entre trajetórias.

Os autores citam ainda que trabalhos anteriores na literatura propuseram diversas maneiras para lidar com problema de similaridade entre trajetórias, como, por exemplo, *Dynamic Time Warping* (DTW), distância *Fréchet*, *Edit Distance on Real Sequence* (EDR), dentre outras. Entretanto, cada uma dessas abordagens captura a similaridade entre trajetórias sob perspectivas distintas, algumas mais focadas na sequência estrutural, enquanto outras consideram apenas a ordem sequencial dos pontos. A desvantagem dessas medidas é que frequentemente exigem muito tempo e recursos computacionais, limitando sua aplicabilidade em grandes conjuntos de dados.

Tendo por base esta problemática, os autores em Yang *et al.* (2021) propuseram um modelo baseado em redes neurais profundas, chamado de *T3S*, que conseguiu criar representações vetoriais (*embeddings*) para cada trajetória analisada em um espaço de baixa dimensão (*d*-dimensional) e adaptável para qualquer cenário, bastando apenas realizar um ajuste simples dos parâmetros prévios ao treino do modelo. O *T3S* foi desenvolvido utilizando a arquitetura de rede neural *Long Short-Term Memory* (LSTM) para processar pontos geoespaciais de trajetórias representadas por células em uma grade, permitindo a incorporação de informações espaciais e estruturais nos vetores aprendidos. Além disso, o modelo *T3S* demonstrou capacidade de acelerar o cálculo de similaridade entre trajetórias, superando outras técnicas de distância e modelos comparados.

3.4 Trajectory Similarity Learning with Auxiliary Supervision and Optimal Matching

O cálculo da similaridade entre trajetórias é um problema fundamental no campo da análise de dados geoespaciais. Assim, identificar a similaridade entre trajetórias tornou-se uma tarefa comum para um amplo número de aplicações reais pertencentes ao cenário tecnológico atual (e.g., análise de tráfego, compartilhamento de caronas, detecção de anomalias, dentre outras). No entanto, o cálculo da similaridade entre trajetórias é uma tarefa de alta complexidade

computacional, tornando-se o principal gargalo para esse domínio de aplicações.

Traj2SimVec (Zhang *et al.*, 2021) é outro modelo projetado para aprender *embeddings* de trajetórias, ao mesmo tempo que visa aproximar funções de similaridade de trajetórias bem conhecidas. O modelo começa com a construção de amostras de treinamento a partir do conjunto de dados original por meio da simplificação de trajetória e da indexação *kd-tree*, possibilitando a geração eficiente de amostras contendo triplas de treinamento (âncora, próximo, distante) a partir de grandes conjuntos de dados, simplificando trajetórias em vetores de comprimento fixo e aproveitando a indexação *kd-tree* para recuperação rápida de pares similares. Um componente crucial do *Traj2SimVec* é seu codificador de trajetória, que usa internamente uma arquitetura baseada em *LSTM*. Assim, as amostras são processadas para produzir incorporações de trajetória, calculando-se distâncias entre subtrajetórias usando programação dinâmica. Este processo garante que o modelo capture semelhanças gerais e de subtrajetórias para um cálculo robusto de similaridade.

Finalmente, durante a fase de aprendizagem, *Traj2SimVec* emprega as funções de perda supervisionadas *Sub-trajectory Distance Loss* e *Trajectory Point Matching Loss*. A primeira aproveita as distâncias entre as subtrajetórias para uma supervisão auxiliar, melhorando a robustez do modelo e capturando relações diferenciadas, enquanto a segunda se concentra em aprender relações ideais de correspondência de pontos entre trajetórias, permitindo a identificação de pontos correspondentes sob várias métricas alvo (nos experimentos, os autores consideraram as métricas *Frechet*, *Hausdorff* e *DTW*).

3.5 Trajectory Semantic Embedding for Living Pattern Recognition in Population

O reconhecimento de padrões comportamentais na vida de uma população é uma tarefa considerada extremamente importante para a tomada de decisões e o planejamento urbano atualmente. Em paralelo, o aumento no volume de informações em aplicações baseadas em geolocalização (e.g., *check-ins* em redes sociais, *e-commerce*, aplicativos de compartilhamento de carona) incorporam com bastante frequência dados importantes sobre localizações sobre pontos de interesse em comum (POI) entre seus usuários.

Informações desse gênero são cruciais para a identificação de padrões/hábitos de vida da população. Com base no exposto, a ferramenta intitulada *habit2vec*, proposta por Cao *et al.* (2020), é um *framework* de aprendizado de máquina não supervisionado que cria representações numéricas (*embeddings*) para os trajetos de usuários. Essas representações conseguem preservar

as informações semânticas dos hábitos de vida de cada usuário envolvido. Para tal, foram utilizados dados históricos de trajetórias da base de dados *Tencent*, conjunto de dados reais de grande escala fornecido por um popular provedor de rede social em Beijing-China, com um total de 123.803 usuários.

A ferramenta consegue reconhecer hábitos de vida típicos, até mesmo, entre usuários que se encontravam em regiões diferentes. Os autores conseguiram isso por meio da incorporação de características temporais no treino da ferramenta além das informações espaciais (*POIs*). Assim, trajetórias distantes espacialmente, mas que seguem um mesmo padrão temporal e/ou compartilham os mesmos *POIs*, conseguem ser identificadas como semelhantes pelas representações vetoriais do *habit2vec*. Deste modo, o *framework* consegue minerar o hábito de vida dos usuários incorporando a semântica espaço-temporal em um mesmo vetor numérico (*embedding*).

3.6 Learning Behavioral Representations of Human Mobility

A mobilidade humana é uma área com enormes desafios que incitam o surgimento de novas aplicações a cada dia. Os autores em Damiani *et al.* (2020) mostraram-se preocupados com os desafios referentes à mobilidade humana e como os dados de trajetórias podem ter um papel relevante na resolução dos problemas deste domínio. Sendo assim, o objetivo principal do trabalho foi definir um grau de similaridade comportamental (*behavioral similarity*) entre indivíduos (pessoas) recorrendo a representações vetoriais obtidas por meio do treinamento de modelos de redes neurais.

Para esse propósito, foram adaptados modelos de representação vetorial considerados estado da arte, previamente utilizados na análise de similaridade semântica entre palavras, com o objetivo de representar dados geoespaciais de trajetórias esparsas. Os dados foram coletados de sistemas de telecomunicação móvel, onde os registros (*CDR – Call Detail Records*) são gerados com base em eventos como o início/fim de chamadas, envio/recebimento de mensagens de texto e *downloads/uploads* de arquivos.

A contribuição principal do trabalho consiste na criação de um novo *framework*, intitulado *mob2vec*, guiado pelo aprendizado representacional de redes neurais, conseguindo generalizar trajetórias para vetores incorporando suas informações comportamentais. A abordagem usa métodos próprios de *PLN* (*Word2Vec* e *Paragraph Vector*) para gerar as representações de *embeddings* para o domínio das trajetórias *CDR*. Assim, o modelo *mob2vec* consegue mapear as trajetórias *CDR* em vetores de dimensões fixas que conseguem preservar os comportamentos de

mobilidade de cada usuário do conjunto de dados analisado.

3.7 Deep Representation Learning of Activity Trajectory Similarity Computation

O aumento exponencial dos dados de trajetória, impulsionado por dispositivos equipados com tecnologia *GPS* e comunicação sem fio, tem sido um foco significativo na pesquisa em bancos de dados espaciais. A problemática da amostragem de trajetória, originada por estratégias distintas de amostragem, impacta negativamente na medição de similaridade. Embora abordagens recentes tenham mitigado esse problema por meio de complementos de trajetória, estas ainda apresentam limitações, ao considerarem predominantemente características espaciais e temporais.

O desenvolvimento de redes sociais *LBSN* (*Location-based Social Network*) proporciona uma nova dimensão da informação, enriquecendo trajetórias com informações semânticas. Com base nisso, o framework *At2vec* (Zhang *et al.*, 2019), propõe uma abordagem que integra características espaço-temporais com dados semânticos de atividades de trajetórias, superando limitações e alcançando resultados superiores em experimentos extensivos com bancos de dados de trajetórias reais. Assim, o *At2vec* pode gerar *embeddings* de trajetórias a partir de trajetórias esparsas de *check-ins*, considerando informações espaciais, temporais e semânticas (atividades). Para isso, o espaço geográfico foi dividido em uma grade, em seguida, o modelo *Word2Vec* foi aplicado para obter incorporações espaço-temporais e o *GloVe* para as incorporações das atividades.

3.8 Modeling Trajectories Obtained from External Sensors for Location Prediction via NLP Approaches

O extenso uso de dispositivos com geolocalização proporciona a construção de bases de dados espaciais contendo grandes volumes de informações de trajetórias. Este tipo de dado apresenta um desafio significativo: prever a próxima localização que um objeto móvel alcançará. Os autores em Cruz *et al.* (2022) reconhecem a complexidade desse problema e buscam determinar os próximos movimentos de uma classe específica de objetos móveis (veículos) com base no histórico de trajetórias obtidas em uma rede de ruas, mapeada por meio de sensores externos de tráfego.

O estudo, denominado *EST Prediction (External Sensor Trajectory Predction)*, avalia

métodos de aprendizado de representação para gerar características espaciais e prever os próximos movimentos dos veículos na rede de ruas. Para isso, a rede é mapeada usando uma grade espacial, permitindo interpretar as trajetórias como sequências de posições de sensores. Cada sensor grava informações suficientes para identificar unicamente cada veículo. Por fim, o modelo *BERT* é empregado para codificar essas trajetórias de sensores externos, gerando representações (*embeddings*) utilizadas para prever o próximo sensor de tráfego onde um objeto móvel passará.

3.9 A Time-aware Trajectory Embedding Model for Next-location Recommendation

A recomendação da próxima localização é uma tarefa essencial para serviços baseados em localização, como, por exemplo, os serviços presentes nas redes sociais. Nestes serviços, a proliferação e o volume dos dados atingem proporções gigantesca, tornando-os uma fonte imensurável de informações que podem ser usadas nas mais variadas tarefas e aplicações no que se refere ao domínio do processamento de dados de mobilidade. Tomando por base essa categoria de dados, Zhao *et al.* (2018) propuseram um modelo de rede neural, intitulado *TA-TEM* (*Time-Aware Trajectory Embedding Model*), voltada para a tarefa de predição/recomendação da próxima localização para usuários de três conjuntos de dados reais de trajetórias esparsas, trajetórias estas formadas pelas posições geoespaciais dos *check-ins* de usuários.

No modelo em questão foram verificados diferentes contextos de trajetórias, pegando as informações espaciais (lon, lat) dos pontos (*check-ins*) e, também, incorporando fatores temporais para auxiliar ainda mais na tarefa de recomendação da próxima localização. Em seguida, os contextos processados foram usados para o treinamento do modelo, resultando na incorporação das localizações em vetores únicos e de tamanho fixo (*embeddings*), ou seja, projetando os pontos das trajetórias em um espaço vetorial de baixa de dimensão, possibilitando que fossem mensurados usando uma simples função de similaridade cosseno.

Para a tarefa proposta, o modelo *TA-TEM* conseguiu integrar múltiplos tipos de contextos de informação de uma maneira unificada e mais resistente a esparsidade dos dados, utilizando a vetorização (*embeddings*) para padrões de dados sequenciais dotados das informações espaciais (*check-ins*) e temporais (*timestamps*). Os experimentos finais demonstraram que o modelo em questão conseguiu uma boa efetividade na predição da próxima localização, superando os demais modelos/técnicas de trabalhos anteriores.

3.10 TraceBERT—A Feasibility Study on Reconstructing Spatial–Temporal Gaps from Incomplete Motion Trajectories via BERT Training Process on Discrete Location Sequences

Em Crivellari *et al.* (2022), os autores reconhecem que dados de trajetórias representam uma fonte essencial de informação para o reconhecimento de padrões de mobilidade humana, desempenhando um papel fundamental em diversos serviços relacionados ao planejamento de transportes. Contudo, esses dados nem sempre são fidedignos quanto ao processo de gravação da informação. Problemas durante a captação dos pontos, como falhas de hardware ou interferência do ambiente, podem resultar em lacunas espaço-temporais e até mesmo na ausência de segmentos nas trajetórias. Essa questão torna-se especialmente crítica em tarefas que requerem informações completas sobre as rotas dos usuários.

Diante desse desafio, a abordagem *TraceBERT* propõe um paralelo técnico com a área do Processamento de Linguagem Natural (PLN). O objetivo central dessa abordagem consiste em introduzir o uso do *Bidirectional Encoder Representations from Transformers (BERT)* no campo de pesquisa de processamento de trajetórias. Em essência, um modelo *BERT* foi treinado para obter representações bidirecionais profundas a partir de sequências de localizações não rotuladas, condicionando seus contextos à esquerda e à direita, visando prever localizações (pontos) ausentes ao longo dos trajetos de usuários. Dessa forma, o *TraceBERT* considera as incorporações (*embeddings*) de trajetórias visando resolver o problema da reconstrução de dados ausentes nestas trajetórias.

3.11 Resumo dos Trabalhos Relacionados

No Quadro 1, é apresentada uma comparação entre os trabalhos relacionados e este trabalho de dissertação. Destaca-se que essa proposta se diferencia dos demais estudos ao explorar alternativas consolidadas no campo do Processamento de Linguagem Natural (PLN) para um novo domínio de aplicação: o campo das trajetórias de objetos móveis. A abordagem visa aprender representações tanto para trajetórias densas quanto esparsas, a fim de atingir um desempenho satisfatório quanto ao cálculo de similaridade espacial entre trajetórias usando essas representações.

Quadro 1 – Comparativo entre os trabalhos relacionados e o trabalho proposto

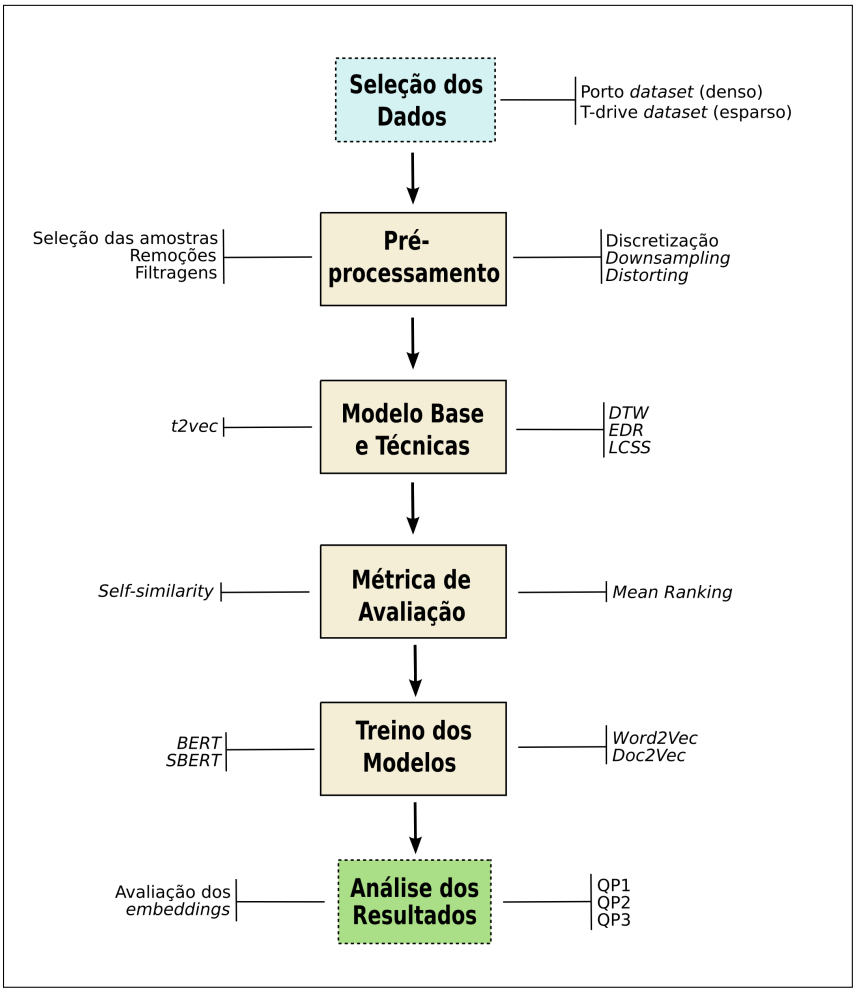
Artigo	Métodos	Conjunto de dados	Discretização	Tarefa
Li *et al.* (2018b)	GRU-based autoencoder + custom loss	Denso e Esparso	Grade de células	Similaridade espacial entre trajetórias
Yao *et al.* (2019)	Attention + LSTM + custom loss	Denso e Esparso	Grade de células	Similaridade espacial entre trajetórias
Yang *et al.* (2021)	Attention + LSTM + custom loss	Denso e Esparso	Grade de células	Similaridade espacial entre trajetórias
Zhang *et al.* (2021)	LSTM + custom loss	Denso	Localizações GPS	Similaridade espacial entre trajetórias
Cao *et al.* (2020)	Word2Vec	Esparso	POIs	Similaridade semântica
Damiani *et al.* (2020)	Word2Vec, Paragraph Vector	Esparso	CDR	Similaridade semântica
Zhang *et al.* (2019)	Word2Vec, GloVe e RNN	Esparso	Grade de células	Similaridade semântica
Cruz *et al.* (2022)	BERT, LSTM	Esparso	Sensores de trânsito	Predição da próxima localização
Zhao *et al.* (2018)	Word2Vec, Paragraph	Esparso	Check-in	Recomendação da próxima localização
Crivellari *et al.* (2022)	BERT	Esparso	CDR	Recuperação de dados faltantes em trajetórias
Este trabalho	**BERT, SBERT Word2Vec, Doc2Vec**	**Denso e Esparso**	**Grade de células**	**Similaridade espacial entre trajetórias**

Fonte: Elaborado pelo autor (2024).

4 PROCEDIMENTOS METODOLÓGICOS

A metodologia deste trabalho compreende seis etapas principais: 1. Seleção dos Dados, 2. Pré-processamento, 3. Modelo Base e Técnicas, 4. Métrica de Avaliação, 5. Treino dos Modelos e 6. Análise dos Resultados. Na Figura 13, é ilustrado o fluxo básico dessas etapas, destacando seus pontos específicos.

Figura 13 – Fluxo metodológico deste trabalho



Fonte: Elaborado pelo autor (2024).

As partes principais dessas etapas serão abordadas nas seções subsequentes deste capítulo e dos capítulos posteriores, destacando-se principalmente a descrição da abordagem utilizada para reaproveitar os modelos de linguagem considerados neste estudo, especificamente para a tarefa de calcular a similaridade espacial entre trajetórias. Inicialmente, neste capítulo, é detalhado o pipeline desenvolvido para o preparo dos dados, tanto para o treinamento quanto para a avaliação desses modelos. A saída desse pipeline consiste em dados de treinamento aumentados e dados de teste aumentados. Posteriormente, é explicado como a fase de treinamento

dos modelos de linguagem foi adaptada, permitindo o treinamento com os dados de treino aumentados.

4.1 Pipeline de Preparação dos Dados para os Modelos de Linguagem

Assumindo que o pipeline de preparação dos dados é aplicado em um conjunto de dados D que já foi pré-processado e dividido nos conjuntos de treino (D_{train}) e teste (D_{test}): o pipeline é responsável por gerar versões aumentadas desses conjuntos, produzindo assim dados adequados para treinar e avaliar modelos de linguagem visando a tarefa do cálculo de similaridade espacial entre trajetórias. É importante observar que o pipeline por si só não é responsável por fazer a discretização das trajetórias.

No presente estudo, o pipeline adotado seguiu o mesmo fluxo de preparação dos dados descrito no artigo do *t2vec* (Li *et al.*, 2018b). Esse pipeline apresenta várias características úteis: (1) permite o fácil reaproveitamento dos modelos de linguagem considerados neste trabalho; (2) considera a variabilidade de amostragem e o ruído, dois fenômenos geralmente encontrados em dados reais de *GPS*, simulando diferentes taxas de amostragem e introduzindo ruído controlado (gaussiano), respectivamente; finalmente, (3) o pipeline aumenta a diversidade dos dados de treinamento para evitar ajuste excessivo e aprimorar a generalização para dados não vistos.

4.1.1 Gerando Dados de Treino Aumentados

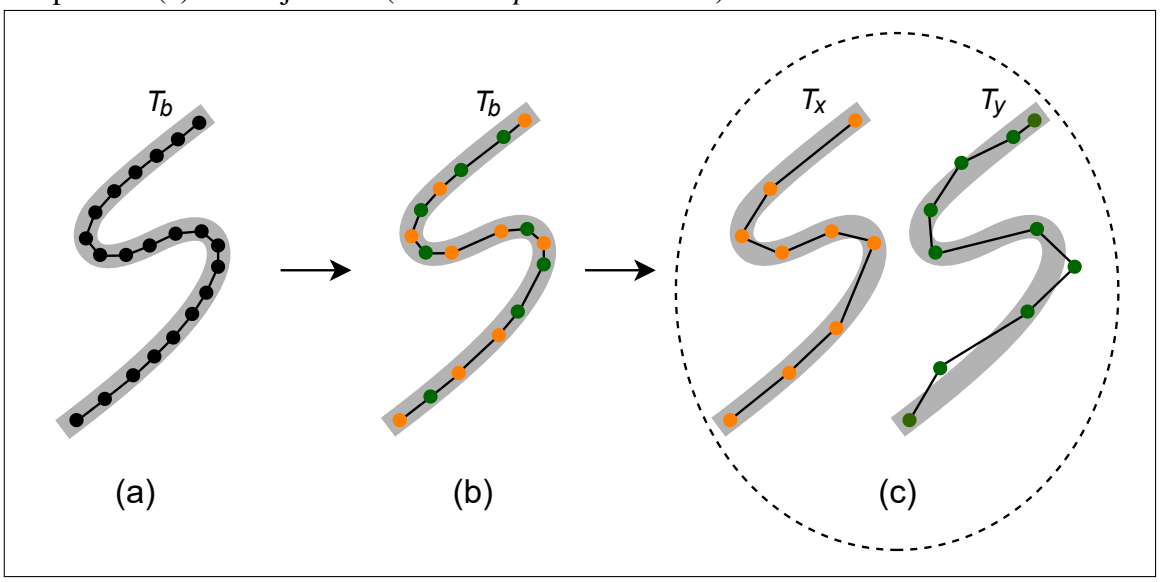
A informação do grau de similaridade entre as trajetórias geralmente não está disponível na maioria das bases de dados reais. Para contornar esse problema, a ideia fundamental aqui consistiu em considerar o conjunto de dados de treino D_{train} e criar sua versão aumentada, criando novas trajetórias semelhantes a partir das trajetórias originais deste conjunto. No entanto, obter variações de trajetórias similares às trajetórias originais não é uma tarefa simples. É preciso garantir que as novas trajetórias sejam parecidas com suas trajetórias originárias. Para isso, semelhante ao *t2vec* (Li *et al.*, 2018b), o princípio básico do aumento dos dados de treino consistiu em criar subtrajetórias com base na rota real R das trajetórias originais, a qual é representada por uma curva continua no espaço que indica o caminho real do objeto móvel de um determinado trajeto, representada pela área sombreada nas trajetórias da Figura 14.

Um dos principais desafios dessa abordagem é a falta de informação sobre a rota real

R nas bases de dados de trajetórias. Para contornar esse problema, Li *et al.* (2018b) mapearam os dados explorando duas observações importantes: (1) tanto as trajetórias com alta taxa amostral, quanto suas variações com taxas amostrais relativamente mais baixas, compartilham a mesma rota real R, e (2) sendo T_b (Figura 14 a) a trajetória original, de maior taxa amostral, ela é mais semelhante à rota real R do que qualquer uma de suas variações de taxa amostral menores, T_x e T_y (Figura 14 c).

Um ponto importante a ser observado é que a rota real R não é disponibilizada na maioria dos conjuntos de dados atuais, inviabilizando a criação das subtrajetórias por esta abordagem. A forma mais plausível encontrada para driblar este problema foi a adoção da trajetória com maior taxa amostral, T_b, como a própria rota real. Assim, na Figura 14, tem-se T_b como a rota real $(T_b = R)$, T_x e T_y como duas variações desta rota real, a primeira obtida pelo processo de redução dos pontos (*downsampling*) e a segunda pelo processo de inclusão de ruído (*distorting*). É notável também que os pontos inicial e final de T_b são mantidos inalterados nas subtrajetórias criadas, isso foi necessário para garantir que as novas trajetórias (T_x e T_y) mantenham o mesmo comprimento da trajetória original (T_b).

Figura 14 – Processo de aumento dos dados de treino. (a) Trajetória original. (b) Mapeamento dos pontos. (c) Subtrajetórias (*downsampled* e *distorted*)



Fonte: Elaborado pelo autor (2024).

Desta forma, para cada trajetória $T \in D_{train}$, o pipeline de aumento dos dados gera um número específico de subtrajetórias com baixa taxa amostral, aplicando o processo de *downsampling*. Em seguida, é aplicado o ruído gausiano a essas subtrajetórias para obter um novo grupo de subtrajetórias, conhecidas como versões distorcidas, por meio do processo de

distorting. Ao final do processo, são produzidos os dados de treinamento aumentados D_{train}^{aug}. As taxas de geração para esses dados serão discutidas no capítulo seguinte.

4.1.2 Gerando Dados de Teste Aumentados

Avaliar a eficácia de diferentes abordagens na avaliação de similaridade para trajetórias é uma tarefa desafiadora devido à ausência de um *ground-truth* (verdade absoluta quanto à similaridade entre todas as trajetórias do *dataset*). Inspirando-se nas comparações de *self-similarity*, auto-similaridade, (Ranu *et al.*, 2015) e *cross-similarity*, similaridade-cruzada, (Su *et al.*, 2013), os autores em Li *et al.* (2018b) introduziram um método de avaliação chamado *most similar search*, que realiza a pesquisa da trajetória mais similar dentro de um imenso conjunto de busca. Este método, uma variação da comparação *self-similarity*, é uma abordagem segura que visa avaliar a qualidade dos *embeddings* aprendidos e, portanto, foi também adotado na avaliação experimental deste trabalho.

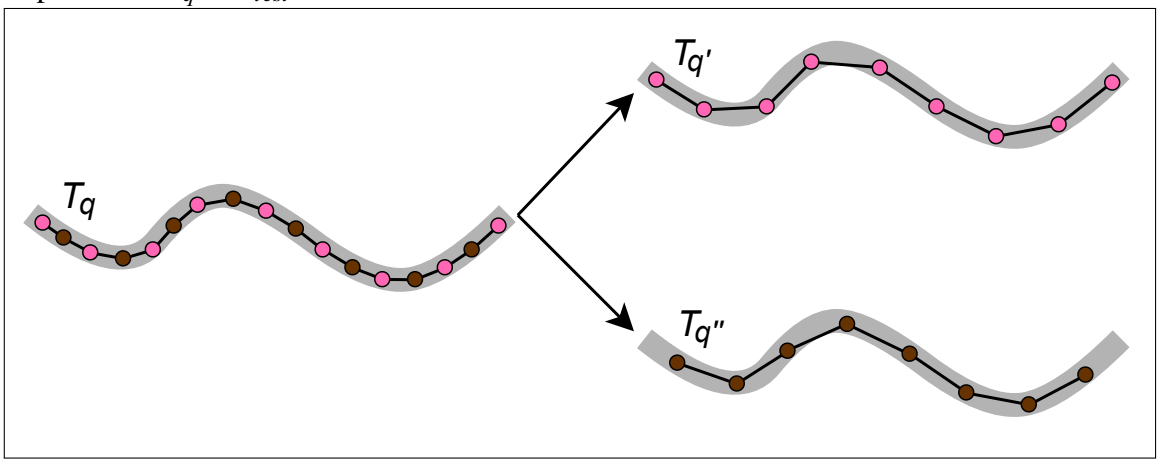
A metodologia segue da seguinte maneira: a partir dos dados de teste D_{test}, é selecionado aleatoriamente um conjunto de trajetórias de consulta, denominado Q. Além disso, um conjunto maior de trajetórias, denotado por B, também é extraído de D_{test} para servir como o conjunto de trajetórias de busca, devendo-se garantir que $Q \cap B = \emptyset$. Então, para cada trajetória $T_q \in Q$ foram geradas duas novas subtrajetórias, $T_{q'}$ e $T_{q''}$, alternado-se seus pontos entre ímpares e pares, como ilustrado na Figura 15. Isso, por sua vez, divide as trajetórias em dois novos subconjuntos: o conjunto de subtrajetórias ímpares Q' e o conjunto de subtrajetórias pares Q''. Este mesmo processo de transformação é aplicado ao conjunto de busca B, produzindo-se B' e B''.

Ao final, Q' é utilizado como o conjunto de trajetórias de consulta, enquanto $S = Q'' \cup B''$ é usado como o espaço de busca. Idealmente, para cada trajetória $T_{q'} \in Q'$ essa abordagem deve reconhecer $T_{q''} \in S$ como a trajetória mais semelhante espacialmente à $T_{q'}$. Os dados de teste aumentados, formados por $Q' \cup S$, foram denotados por D_{test}^{aug}. Detalhes adicionais sobre como Q' e S foram utilizados para avaliar quantitativamente a qualidade dos *embeddings* gerados pelos modelos de linguagem são explicados no Capítulo 5.

4.1.3 Treinando Modelos de Linguagem com os Dados de Treino Aumentados

Ao reutilizar modelos de linguagem de modo a torná-los treináveis em dados de trajetória aumentados, uma questão importante é converter as trajetórias em um tipo de sequência

Figura 15 – Aumento dos dados de teste: criação das duas subtrajetórias $T_{q'}$ e $T_{q''}$ alternando os pontos de $T_q \in D_{test}$.



Fonte: Elaborado pelo autor (2024).

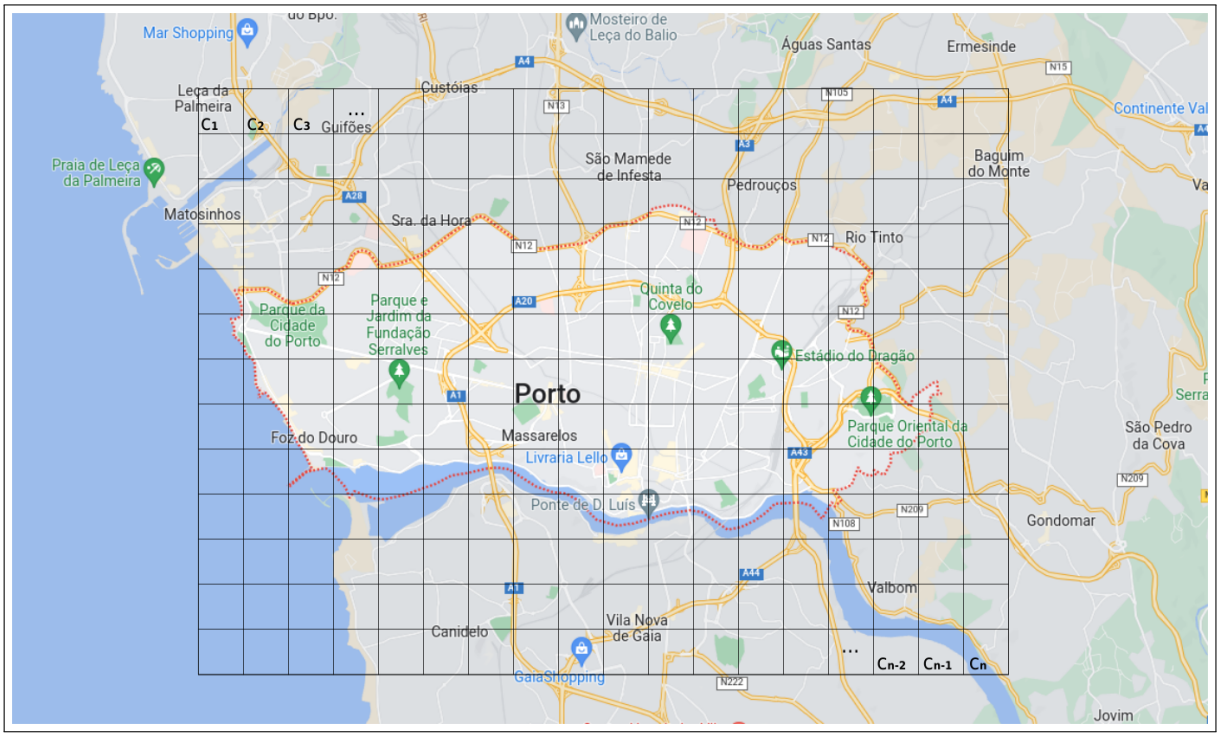
que esses modelos possam compreender. Para resolver isso, foi implementada a técnica de discretização em grade baseada na abordagem *Realm* (Guting; Schneider, 1995). Primeiramente, a área geográfica onde as trajetórias de D^{aug}_{train} se movem foi mapeada usando a grade uniforme C, particionando-se toda a região em unidades de representação menores, denominadas células c_i. A Figura 16 sintetiza como esta grade virtual pode ser estabelecida para mapear a região da cidade de Porto, Portugal, em unidades menores (células). Ainda na imagem, é possível notar que as células possuem tamanho fixo (altura x largura) e seguem uma ordem crescente quanto a numeração dos seus identificadores (*tokens*). Neste trabalho, tanto para o conjunto de dados Porto quanto para o conjunto T-drive, foi utilizada uma grade C com dimensão de $50x40$ km, onde as células têm um tamanho fixo de 100 m de largura e altura.

Posteriormente, tendo a grade C bem definida, as trajetórias em D^{aug}_{train} e D^{aug}_{test} são convertidas de uma sequência de pontos geoespaciais para uma sequência de células, ou seja, sequência *tokens* correspondentes que identificam unicamente cada célula dentro grade C. Este processo, em última análise, transforma os dados de trajetória aumentados em dados de trajetória aumentados discretizados, produzindo os dados de treinamento aumentados e discretizados D'^{aug}_{train} e os dados de teste aumentados e discretizados D'^{aug}_{test}. Agora, é possível explicar como o treinamento de cada um dos modelos de linguagem selecionados foi realizado usando D'^{aug}_{train}.

4.2 Treino dos Modelos de Linguagem

Esta etapa marca o momento em que os modelos de linguagem selecionados (*Word2Vec*, *Doc2Vec*, *BERT* e *SBERT*) recebem efetivamente os dados de treinamento, que foram aumentados

Figura 16 – Exemplo da grade uniforme C sobre a cidade de Porto, Portugal



Fonte: Elaborado pelo autor (2024).

e discretizados, representados por $D_{train}^{'aug}$. Aqui, serão abordadas as características de treinamento específicas de cada modelo, destacando as arquiteturas utilizadas e os formatos dos dados de entrada.

4.2.1 Treinamento do Word2Vec

O modelo *Word2Vec*[1] (Mikolov *et al.*, 2013) é capaz de projetar representações de palavras em um espaço vetorial qualquer $\mathbb{R}^n$. Neste trabalho, o *Word2Vec* foi treinado do zero usando os dados de treinamento aumentados e discretizados $D_{train}^{'aug}$. Desta forma, esses dados de treino atuaram como o corpus do modelo, enquanto as células da grade transpassadas pelas trajetórias representaram o vocabulário. À medida que o modelo foi treinado, ele empregou a arquitetura *Continuous Bag of Words (CBOW)* para aprender os *embeddings* de cada célula c pertencente ao vocabulário V. Este processo envolveu predizer uma célula alvo dentro de uma trajetória discretizada com base no seu contexto de células vizinhas. O modelo, então, ajustou iterativamente os *embeddings* das células para minimizar o erro de predição, capturando, desse modo, os contextos espaciais definidos pelas sequências de *tokens* das células.

O objetivo deste trabalho é, em última instância, calcular similaridade espacial entre

[1] https://radimrehurek.com/gensim/models/word2vec.html

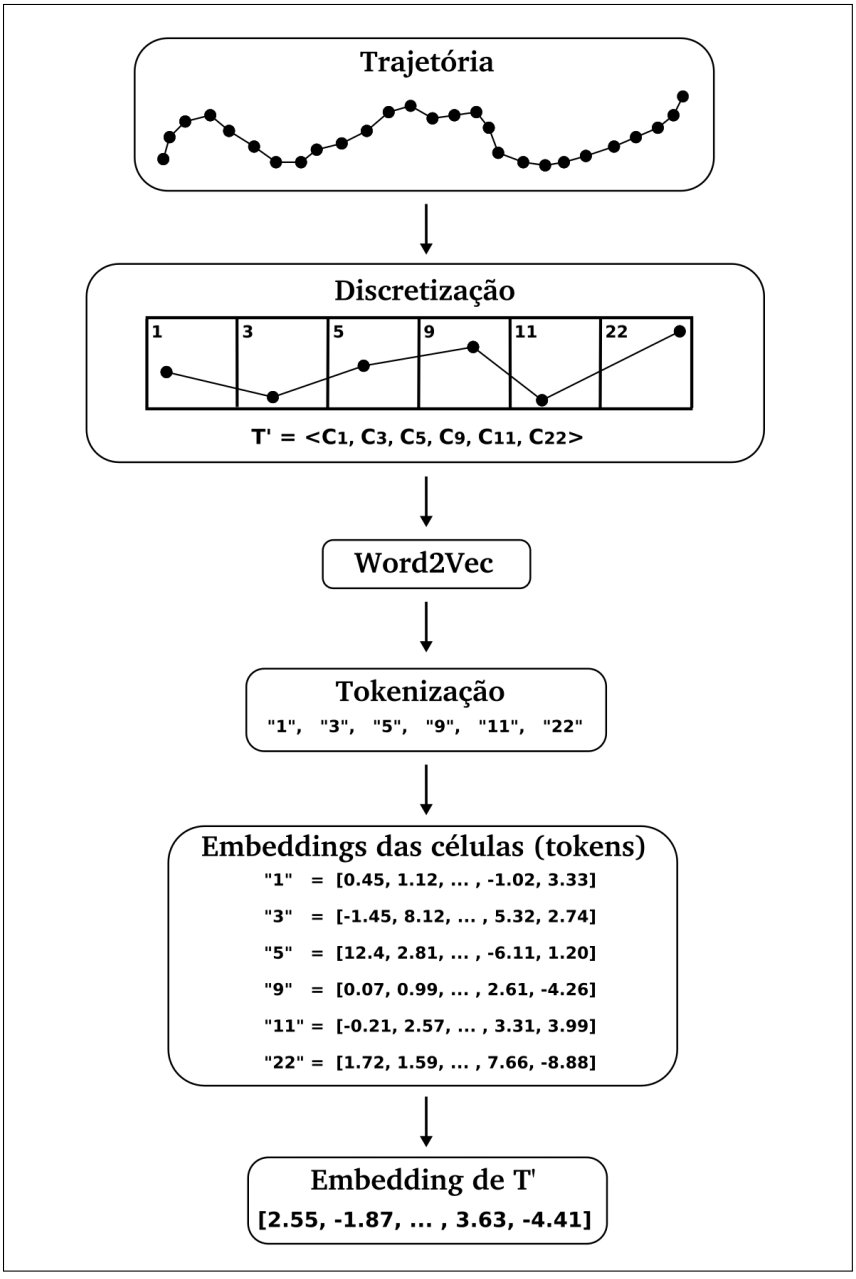
trajetórias usando *embeddings*. Para obter esses *embeddings*, após concluir o treinamento do *Word2Vec* com os dados de treino D'^{aug}_{train}, os *embeddings* das trajetórias de teste D'^{aug}_{test} foram calculados por meio da média dos *embeddings* das células que representam cada trajetória. Assim, sendo $T' = \langle c_1, \ldots, c_n \rangle$ alguma trajetória de teste discretizada, de comprimento n, e $W_{T'} = \langle w_1, \ldots, w_n \rangle$ sua sequência correspondente de *embeddings* das células. Então, o *embedding* de T' é computado como o *embedding* médio (uma escolha comum na literatura, e.g., Cruz *et al.* (2022)) dos *embeddings* das células atravessadas por T. Assim, dado uma trajetória T, seu *embedding* (v^T) pode ser obtido conforme a Equação 4.1.

$$v^T = \frac{1}{|W_{T'}|} \sum_{w \in W_{T'}} w. \tag{4.1}$$

Como argumentado anteriormente, para que o modelo *Word2Vec* pudesse tratar dados de trajetórias, as informações geoespaciais dos pontos (long, lat), análogo às informações textuais, precisaram passar por um processo de discretização para poderem ser compreensíveis pelos modelos de linguagem, mapeando pontos em células de uma grade e, por sua vez, células em *tokens*. A Figura 17 fornece uma visão mais clara do processo, demonstrando o fluxo ao utilizar o modelo *Word2Vec* previamente treinado para obter o *embedding* de uma trajetória de teste T pertencente a D^{aug}_{test}.

Na figura, o primeiro retângulo contém a trajetória de teste bruta T, formada por pontos geoespaciais (lon, lat), que será vetorizada (obtido o *embedding*) no final do processo. No segundo retângulo, ocorre o mapeamento dos pontos da trajetória para as células correspondentes, ou seja, a discretização. Ainda no segundo retângulo, são extraídos os *tokens ids* que juntos representam a trajetória no seu formato sequencial de células. No quarto retângulo, a sequência obtida é tokenizada utilizando o tokenizador do modelo. No quinto retângulo, usando o modelo *Word2vec* treinado, os *embedding* de cada célula são inferidos. Por fim, o *embedding* da trajetória completa, v^T, é obtido calculando-se a média vetorial dos *embeddings* de todas as células que compõem trajetória completa discretizada, T'.

Figura 17 – Representação vetorial (*embedding*) da trajetória usando *Word2Vec*



Fonte: Elaborado pelo autor (2024).

4.2.2 Treinamento do Doc2Vec

Na busca por gerar *embeddings* para trajetórias inteiras, semelhante à forma como as sequências textuais são tratadas, foi considerado um modelo capaz de aprender a partir de estruturas semelhantes a sentenças, *Doc2Vec*[2] (Le; Mikolov, 2014). Ao contrário do modelo *Word2Vec*, que requer uma representação vetorial para cada célula do vocabulário, o *Doc2Vec*, por ser projetado para lidar com sentenças inteiras, consegue inferir um vetor (*embedding*)

[2] https://radimrehurek.com/gensim/models/doc2vec.html

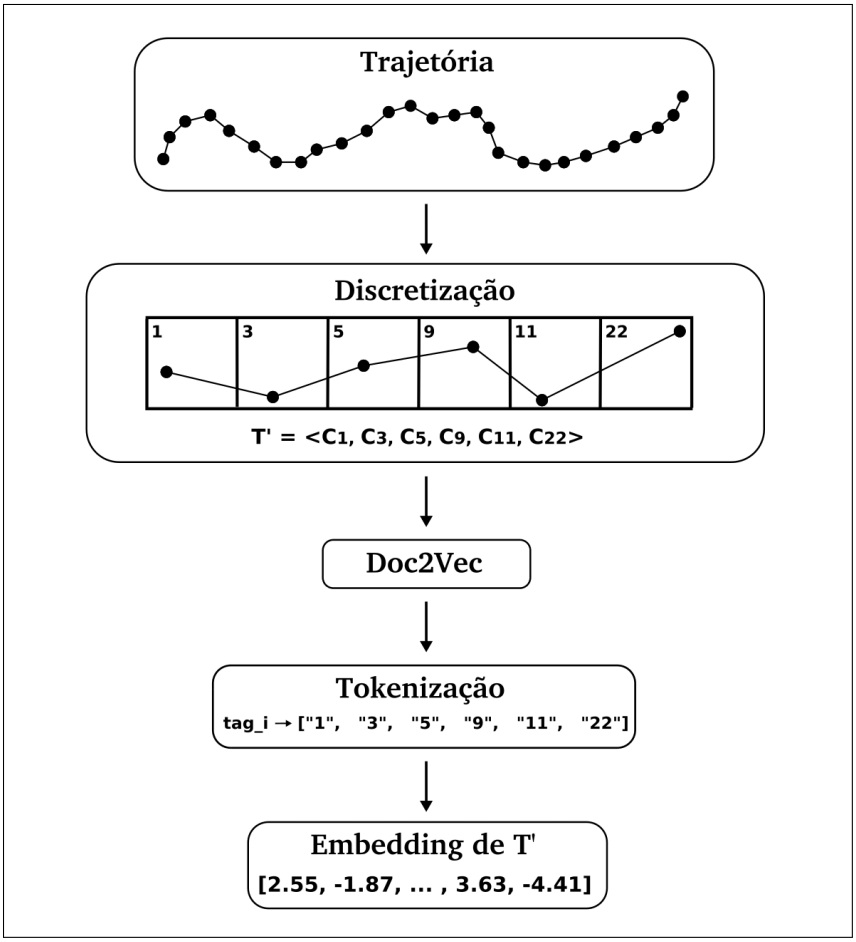
diretamente para a trajetória completa. Isso intuitivamente significa que, durante o treinamento do modelo, as relações espaciais entre os pontos podem ser capturadas de maneira mais eficiente, pois o *Doc2Vec* considera a estrutura completa da trajetória, em vez de tratar apenas células individuais.

As representações das trajetórias foram treinadas conforme as duas principais variantes do *Doc2Vec*: PV-DM (*Distributed Memory*) ou PV-DBOW(*Distributed Bag of Words*). Na primeira variante, cada documento (no caso deste trabalho, sequência de células que representa a trajetória T, ou seja, a trajetória discretizada T') foi inicialmente representado por um vetor de dimensão fixa, sendo inicializado aleatoriamente. Durante o treinamento, a tarefa do modelo foi prever a próxima célula dentro do contexto de células na sentença. Assim, o modelo vai ajustando os pesos do *embedding* da sequência para minimizar a perda (erro) na previsão da célula seguinte. Ao final do treinamento, o vetor médio da sequência aprendido é utilizado como a representação vetorial da trajetória.

Na segunda variante do *Doc2Vec*, a tarefa do modelo não é prever a próxima palavra, mas sim aprender a representação vetorial do documento inteiro. Neste trabalho, o conjunto de sentenças que representam as trajetórias completas $D_{train}^{'aug}$ foi fornecido como entrada para o modelo, e o modelo gerou a representação vetorial para cada um dessas sentenças (trajetórias T'), ignorando a ordem que as células ocorriam nas trajetórias. Isso segue o princípio do "saco de palavras" (*bag of words*), em que a ordem das palavras não é considerada em modelos de processamento de linguagem natural. Na Figura 18, é ilustrado o processo de vetorização de uma trajetória teste utilizando o modelo *Doc2Vec* já treinado.

Os três primeiros retângulos são idênticos ao fluxo do *Word2Vec*, representando a trajetória bruta, trajetória discretizada e o modelo. A principal diferença ocorre a partir do quarto retângulo, onde os *ids* da sentença são tokenizados e marcados com um identificador único (tag_i) para cada trajetória T_i', possibilitando a identificação única da trajetória pelo modelo. Por fim, o modelo consegue inferir diretamente uma representação vetorial (*embedding*) para a trajetória completa T, como ilustrado no quinto retângulo da figura em questão.

Figura 18 – Representação vetorial (*embedding*) da trajetória usando *Doc2Vec*



Fonte: Elaborado pelo autor (2024).

4.2.3 Treinamento do BERT

Modelos baseados em contextualização, como o *BERT*[3] *(Bidirectional Encoder Representations from Transformers)* (Devlin *et al.*, 2019), proporcionaram uma compreensão detalhada dos diversos contextos nos quais uma palavra pode aparecer. Isso permitiu criar representações (*embeddings*) diferentes de uma mesma palavra, dependendo do contexto circundante. Por exemplo, nas sentenças “sentou no banco da praça” e “sacou dinheiro do banco”, a palavra “banco” terá *embeddings* diferentes em cada uma das frases. Isso diferencia o *BERT* de modelos não contextualizados, como o *Word2Vec*, nos quais cada palavra possui apenas uma representação estática, independente do contexto.

Neste estudo, propôs-se o emprego do modelo *BERT* para a avaliar a similaridade espacial entre trajetórias da seguinte forma: utilizou-se a tarefa de Modelo de Linguagem Mascara (MLM), do termo inglês *Masked Language Model*, para treinar o *BERT* em *embeddings*

[3] https://huggingface.co/docs/transformers/model_doc/bert

de células. Nessa configuração, 15% dos *tokens* das células em cada trajetória discretizada $T' \in D'^{aug}_{train}$ foram aleatoriamente mascarados. O modelo foi então treinado do zero (*from scratch*) para prever esses *tokens* mascarados com base no contexto circundante dentro das trajetórias.

Após concluir o treinamento do modelo, os *embeddings* das trajetórias de teste D'^{aug}_{test} foram calculados inserindo as trajetórias discretizadas no modelo *BERT* treinado. A última camada do modelo produziu uma sequência de *embeddings* de células correspondentes a cada *token* de célula na trajetória. Posteriormente, de maneira semelhante à abordagem utilizado com o *Word2Vec*, o *embedding* final da trajetória foi gerado calculando-se a média dos *embeddings* das células que compõem tal trajetória.

Desta forma, o modelo treinado conseguiu gerar representações espacialmente contextualizadas para as células de entrada, além de prever células mascaradas em novas trajetórias. No entanto, dado que a tarefa principal consistia na detecção de similaridade espacial entre trajetórias, o modelo treinado teve que gerar uma representação vetorial (*embedding*) para cada trajetória completa dos dados de teste, como argumentado anteriormente. O processo de geração desses *embeddings* segue o mesmo fluxo apresentado na Figura 17, com a diferença crucial de que o terceiro retângulo deve representar o modelo *BERT*.

4.2.4 Treinamento do SBERT

O modelo *Sentence BERT (SBERT)*[4] permite treinar representações de *embeddings* com base em modelos *BERT* pré-treinados. Diferentemente de modelos que consideram apenas palavras isoladas, o *SBERT* destaca-se ao considerar a contextualização semântica de toda a sentença, permitindo uma compreensão mais profunda do significado. O processo de treinamento do *SBERT* envolve a carga de um modelo *BERT* pré-treinado em grandes conjuntos de dados, seguido pelo ajuste fino para uma tarefa específica. Essa abordagem torna o *SBERT* uma ferramenta flexível, capaz de ser adaptada para diversas aplicações, aproveitando a riqueza de informações presentes em grandes conjuntos de dados.

Considerando que neste trabalho o modelo *BERT* foi pré-treinado com os dados de treinamento aumentados e discretizados D'^{aug}_{train}, o modelo *BERT*, então, foi carregado no *SBERT* e configurou-se o ajuste fino para a tarefa de avaliar a similaridade espacial entre trajetórias. Quanto a elaboração dos dados de entrada do modelo, foram criadas triplas de treinamento do tipo (T_b, T_x, α), onde α representa o grau de similaridade entre uma trajetória de alta taxa amostral T_b

[4] https://www.sbert.net/docs/package_reference/SentenceTransformer.html

e sua variação de baixa taxa amostral T_x. No próximo capítulo, a abordagem usada para a criação dos trios de treino do *SBERT* será discutida, incluindo a definição dos graus de similaridade adotados. O processo de geração dos *embeddings* segue o mesmo fluxo apresentado na Figura 18, com a diferença crucial de que o terceiro retângulo deve representar o modelo *SBERT* e o quarto não utiliza a informação *tag_i*, usada para identificar unicamente cada documento.

5 EXPERIMENTOS E RESULTADOS

Esta seção descreve a configuração experimental do presente trabalho, detalhando os conjuntos de dados de trajetória considerados (Seção 5.1) e como eles foram inicialmente pré-processados (Seção 5.2). Em seguida, apresenta as diversas abordagens concorrentes consideradas na avaliação experimental (Seção 5.3) e como os dados de treino foram preparados (Seção 5.4). Em seguida, é destacado como as abordagens baseadas em aprendizado profundo foram treinadas (Seção 5.5), seguido pela avaliação de eficácia das mesmas (Seção 5.6). Por fim, detalha como os diversos concorrentes foram avaliados na tarefa de avaliação de similaridade espacial entre trajetórias (Seção 5.7).

5.1 Seleção dos Conjuntos de Dados (*Datasets*)

Visando tratar um escopo real de dados geoespaciais, o presente trabalho utiliza trajetórias de *GPS* de dois conjuntos de dados públicos: Porto e T-drive. Ambos são conjuntos de dados de trajetórias obtidos via captação ativa da frota de táxis das cidades de Porto, Portugal, (Porto *dataset*) e Beijing, China, (T-drive *dataset*), sendo o primeiro considerado um conjunto com trajetórias de perfil denso ($sr \leq 15$ segundos) e o segundo considerado um conjunto com trajetórias de perfil esparso ($sr > 15$ segundos).

O conjunto de dados Porto abrange um período de 19 meses e inclui 1.7 milhões de trajetórias registradas por 442 táxis equipados com dispositivos *GPS*. Esses táxis reportaram suas localizações a cada 15 segundos, em média, percorrendo uma distância média de 130,4 metros entre amostras consecutivas. Em contraste, o conjunto de dados T-drive foi coletado durante a primeira semana de fevereiro de 2008, compreendendo 10.357 trajetórias. No entanto, este conjunto possui um intervalo médio mais longo, de 187 segundos entre amostras, resultando em uma distância média maior, 4,2 quilômetros, entre as amostras. Isso produz trajetórias consideravelmente mais esparsas em comparação com as do conjunto Porto. A Tabela 1 resume as principais configurações dos dois conjuntos de dados, ilustrando suas características originais antes da aplicação da etapa de pré-processamento.

5.2 Pré-processamento

Dadas as distintas metodologias de coleta e características dos conjuntos de dados Porto e T-drive, etapas específicas de pré-processamento foram adaptadas para cada um. Essas

Tabela 1 – Estatísticas dos conjuntos de dados (*datasets*)

Conjunto de Dados	Total de Pontos	Número de Trajectórias	Taxa média de capatação	Distância média entre pontos
Porto	74,269,739	1,710,670	15 s	130.4 m
T-drive	15,117,824	10,357	187 s	4.2 km

Fonte: Elaborado pelo autor.

etapas de pré-processamento foram cruciais para refinar cada conjunto de dados, garantindo que as trajetórias usadas nos experimentos fossem representativas e propícias para o treinamento e avaliação precisa dos modelos. Observa-se que, após o pré-processamento, ambos os conjuntos de dados foram divididos em conjunto de treinamento (70%) e conjunto de teste (30%).

5.2.1 Pré-processamento do Conjunto de Dados Porto

Para o conjunto de dados do Porto, que apresenta trajetórias densas, o passo inicial envolveu a remoção de trajetórias com menos de 30 pontos, eliminando trajetórias mais curtas e com menos informação. Esta ação reduziu o tamanho do conjunto de dados de 1.7 para 1.2 milhões de trajetórias. Em seguida, foram realizadas remoções de dados que apresentavam ruídos, muitas vezes causados por lacunas/inconsistências nos sinais de *GPS*, geralmente ocasionadas em áreas com geografia desafiadora (i.e., áreas montanhosas ou cânions urbanos).

5.2.2 Pré-processamento do Conjunto de Dados T-drive

Assim como no conjunto de dados Porto, a primeira etapa de pré-processamento foi a remoção das trajetórias com poucos pontos, por se tratar de um conjunto de dados esparsos, aquelas com menos de 5 pontos foram removidas. Outra etapa do pré-processamento foi a remoção de ruídos e pontos de paradas. Além disso, o conjunto de dados T-drive possui certas particularidades, por exemplo, há muitas trajetórias que a gravação dos dados duraram todos os sete dias de coleta, resultando em trajetos bastante longos, como a trajetória de id 6275, que teve um comprimento total de 154.688 pontos. Para contornar esse problema, uma etapa adicional de pré-processamento foi realizada, a segmentação das trajetórias que possuíam intervalos entre pontos adjacentes maiores que 300 segundos (5 minutos), possibilitando assim, uma melhor representação dos dados e um aumento no número de amostras das trajetórias esparsas.

5.3 Abordagens Concorrentes

Na avaliação experimental, uma variedade de métodos foram comparados, abrangendo três categorias distintas: Abordagens Clássicas, Arquitetura Referência de Aprendizado Profundo para o Cálculo de Similaridade Espacial entre Trajetórias e Modelos de Linguagem (selecionados pelo presente trabalho).

5.3.1 Abordagens Clássicas

Três métodos clássicos foram incluídos para avaliar a similaridade espacial entre trajetórias, todos eles baseados em programação dinâmica. Esses métodos serviram como *baselines* de métodos clássicos e incluem: *Dynamic Time Warping (DTW)* (Kruskal, 1983), *Edit Distance on Real Sequences (EDR)* (Levenshtein, 1965), e *Longest Common Subsequence (LCSS)* (Shuncheng *et al.*, 2019). A *DTW* busca o alinhamento ótimo entre duas séries temporais, permitindo uma comparação abrangente. A *EDR* ajusta duas sequências reais (trajetórias) de modo a quantificar a semelhança entre elas considerando operações de edição (inserção, exclusão e substituição). A *LCSS* identifica correspondência com base em um limiar de similaridade (maior subsequência de pontos em comum entre duas trajetórias), oferecendo uma solução personalizada para a avaliação da similaridade espacial entre trajetórias.

5.3.2 Arquitetura Referência de Aprendizado Profundo para o Cálculo de Similaridade Espacial entre Trajetórias

Como arquitetura de referência para avaliação experimental, optou-se pelo modelo *t2vec* (Li *et al.*, 2018b), um modelo de aprendizado de representações (*embeddings*) projetado para ser robusto quanto ao cálculo de similaridade para trajetórias com baixas taxas amostrais, não uniformes e sujeitas a ruídos.

5.3.3 Modelos de Linguagem

Os modelos de linguagem adotados, como mencionado no Capítulo 2, foram *Word2Vec*, *Doc2Vec*, *BERT* e *SBERT*.

5.4 Preparação dos Dados de Treino Aumentados

Os métodos clássicos, sendo comparações algorítmicas, não necessitam da fase de treinamento, portanto, o que é explicado a seguir não se aplica a eles. É importante relembrar que, para os modelos de linguagem, o conjunto de treinamento inicial D_{train} é primeiramente aumentado, obtendo-se D^{aug}_{train} (como exposto na seção 5.4). As trajetórias em D^{aug}_{train} foram discretizadas de acordo a grade uniforme C, e apenas aquelas células que pertenciam ao vocabulário (*hot cells*) foram mantidas. Isso produziu os dados de treinamento aumentados e discretizados D'^{aug}_{train} usados no treinamento dos modelos de linguagem.

Em relação ao *t2vec*, conforme detalhado em (Li *et al.*, 2018b), este autoencoder, baseado em *Gated Recurrent Unit (GRU)*, visa reconstruir uma trajetória original $T \in D_{train}$ a partir de sua contraparte distorcida e com amostragem reduzida $T^{aug}_{train} \in D^{aug}_{train}$. Similarmente aos modelos de linguagem, o *t2vec* trabalha com trajetórias discretizadas (i.e., sequências de *tokens*). Para tal, D'^{aug}_{train} deve passar por uma transformação adicional: cada trajetória $T'^{aug}_{train} \in D'^{aug}_{train}$ tem que ser emparelhada com a versão discretizada da trajetória original $T \in D_{train}$ a qual T'^{aug}_{train} foi derivada. Este emparelhamento forma uma nova versão dos dados de treinamento aumentados e discretizados, onde cada exemplo consiste em um par (*trajetória_variada*, *trajetória_original*), ou seja, (T'^{aug}_{train}, T'). Assim, o *t2vec* pode ser treinado usando esses pares.

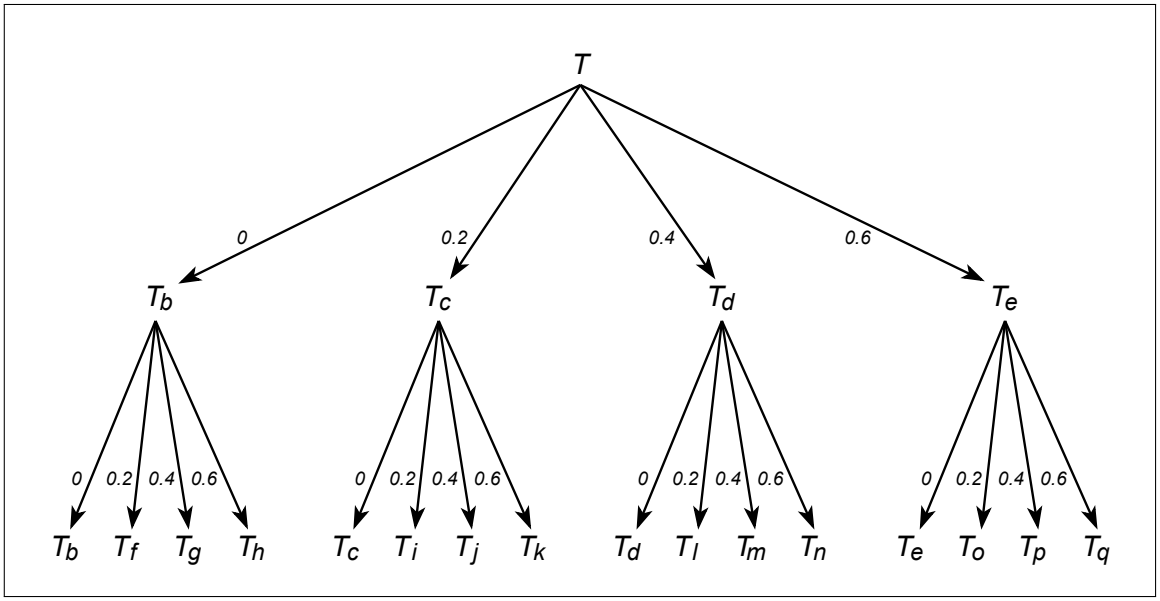
Para ambas as classes de modelos, D^{aug}_{train} foi gerado usando as seguintes taxas $\{0, 0.2, 0.4, 0.6\}$ para a redução de pontos (*downsampling*). Subsequentemente, as taxas de distorção (*distorting*) $\{0, 0.2, 0.4, 0.6\}$ foram aplicadas para obter as versões de trajetórias perturbas por meio do ruído gaussiano. Ao final, para cada trajetória $T \in D_{train}$, 16 diferentes versões (*downsampled* e *distorted*) foram criadas em D^{aug}_{train}, conforme ilustrado na Figura 19. Por fim, as células da grade uniforme C foram configuradas com um tamanho de 100 m, e uma célula passava a ser considerada no vocabulário (ou seja, considerada uma *hot cell*) se pelo menos 50 amostras de pontos em D^{aug}_{train} caíssem nela.

5.5 Treino dos Modelos Baseados em Aprendizado Profundo

A seguir, são esclarecidos os detalhes de como os modelos de baseados em aprendizado profundo, i.e., *t2vec* e os modelos de linguagem selecionados, foram treinados.

t2vec. O treinamento deste modelo foi realizado seguindo as diretrizes dos autores em Li *et al.* (2018b), utilizando uma versão ligeiramente modificada dos dados de treino aumentados

Figura 19 – Criando versões da trajetória T. Segundo nível da árvore representa as trajetórias *downsampled*. Terceiro nível representa as trajetórias *downsampled* e *distorted*



Fonte: Elaborado pelo autor (2024).

e discretizados apresentados na seção 5.4. O modelo utiliza internamente duas *GRUs* de três camadas, uma no codificador e a outra para o decodificador. Os hiperparâmetros ideias para o *t2vec* estão descritos na Tabela 2. As camadas das duas *GRUs* possuem o número de unidades das camadas internas (i.e., *hidden_size*) igual a 256. *embedding_size* indica a dimensionalidade dos *embeddings* das trajetórias, que foi setado para 256. Uma janela de contexto, *window_size*, de 10 células foi usado para cada aprendizado de representação das células. O parâmetro *learning_rate* controla as atualizações de pesos durante a otimização. Finalmente, para otimizar o treinamento, o *t2vec* emprega a função de perda aproximada espacial (*approximate spatial proximity-aware loss function*) focando as k células mais próximas (*k-nearest cells*) de uma célula alvo, em vez de todo o vocabulário, melhorando assim a eficiência computacional.

Tabela 2 – Hiperparâmetros do *t2vec*

Hiperparâmetros	**Valores de referência**
hidden_size	256
embedding_size	256
window_size	10
learning_rate	0,001
k-nearest cells	20

Fonte: Elaborado pelo autor.

Word2vec. Treinado com os dados aumentados e discretizados D'^{aug}_{train} conforme os procedimentos explanados na subseção 4.1.3, o modelo *Word2Vec* foi experimentado com vários tamanhos

de vetores de incorporação (*embedding_size*), que determina o número de neurônios internos do modelo, e de janelas de aprendizado (*window_size*). A arquitetura usada foi a *CBOW* (hiperparâmetro *sg*) devido a sua eficiência em predizer uma palavra alvo com base em seu contexto. Além disso, o modelo excluiu as palavras (células) que apareciam menos que a frequência permitida (*min_count*), melhorando assim a eficiência do treinamento. Finalmente, o modelo foi treinado usando um número de épocas variando de 1 a 1.000. A Tabela 3 mostra todos os hiperparâmetros testados, com os valores em negrito representando a melhor configuração de hiperparâmetros.

Tabela 3 – Hiperparâmetros do *Word2Vec*

Hiperparâmetros	**Valores de referência**
embedding_size	$\{16, 32, \mathbf{64}, 128, 256, 512\}$
window_size	$\{4, \mathbf{5}, 6, 8, 10\}$
sg	CBOW
min_count	3
epochs	$\{1, 2, \mathbf{3}, \ldots, 1000\}$

Fonte: Elaborado pelo autor.

Doc2vec. Assim como o *Word2Vec*, o *Doc2Vec* foi treinado com os dados $D_{train}^{'aug}$ com uma série de valores de *embedding_size*. Além disso, o modelo foi treinado com vários valores de *n_grams*: hiperparâmetro que determina o tamanho da sequência de palavras (no caso deste trabalho, sequência de células) que modelo irá considerar no aprendizado dos *embeddings*. A arquiterua selecionado foi a *PV-DM*, por sua efetividade em aprender documentos contextualizados. Da mesmo forma que o *Word2Vec*, o hiperparâmetro *min_count* trava uma limite para a frequência mínima de palavras no corpus a serem incluídas no vocabulário. Por último, o modelo foi treinado em número de épocas variando também de 1 a 1.000. Na Tabela 4, é ilustrado todos os hiperparâmetros testados, destacando-se em negrito a melhor configuração de hiperparâmetros para este modelo.

Tabela 4 – Hiperparâmetros do *Doc2Vec*

Hiperparâmetros	**Valores de referência**
embedding_size	$\{16, 32, \mathbf{64}, 128, 256, 512\}$
n_grams	$\{4, \mathbf{5}, 6, 8, 10\}$
dm	1 (PV-DM)
min_count	3
epochs	$\{1, \ldots, \mathbf{30}, \ldots, 1000\}$

Fonte: Elaborado pelo autor.

BERT. Este modelo foi treinado também com os dados $D_{train}^{'aug}$ conforme os procedimentos na subseção 4.1.3. A Tabela 5 ilustra todos os hiperparâmetros utilizados, com os valores em negrito representando a melhor configuração. No treinamento, foi explorado uma variedade de valores de *hidden_size* e configurações *transformers* para aprender os *embeddings* dos *tokens* (que, na subseção 4.1.3, são usados para calcular o *embedding* da trajetória final). Mais precisamente, *num_hidden_layers* determina quantas dessas camadas estão empilhadas umas sobre as outras, *num_attention_heads* determina quantos mecanismos de atenção distintos operam em paralelo dentro de cada camada *transformer*, e *max_position_embeddings* determina o comprimento máximo das sequências de entrada que o modelo pode processar. Finalmente, o treinamento é conduzido usando um número de *epochs* variando de 1 a 10.

Tabela 5 – Hiperparâmetros do *BERT*

Hiperparâmetros	**Valores de referência**
hidden_size	$\{32, 64, 128, 256, 512, 1024, 2048\}$
num_hidden_layers	$\{\mathbf{6}, 12\}$
num_attention_heads	$\{8, \mathbf{16}\}$
max_position_embeddings	$\{\mathbf{512}, 1024\}$
epochs	$\{\mathbf{1}, \ldots, 10\}$

Fonte: Elaborado pelo autor.

SBERT. O treinamento deste modelo foi dado inicialmente pelo carregamento do modelo *BERT* pré-treinado que teve a melhor configuração de parâmetros mencionada no tópico anterior. Logo após ao carregamento do melhor modelo *BERT* pré-treinado pelo *SBERT*, o próximo passo foi a criação das triplas de treinamento mencionadas na subseção 4.2.4. Estes triplas foram criadas da seguinte forma: para cada trajetória $T \in D_{train}$, suas 16 variações em D_{train}^{aug} foram mapeadas em triplas do tipo (T, T_x, α), onde T representa a trajetória original, T_x a variação de T e α define o grau de similaridade entre T e T_x. O grau máximo de α foi representado por 1 ($\alpha = 1$) e o grau mínimo por 0 ($\alpha = 0$). Como as subtrajetórias são sempre variações semelhantes à trajetória original T, seguindo um padrão de similaridade da mais similar a menos similar, 16 triplas de treino foram criadas, cada um com seu valor α específico. Assim, baseado na Figura 19, para cada trajetória original T_b, foram criadas as 16 seguintes triplas de treinamento, da mais similar a menos similar: $\{(T_b, T_b, 1), (T_b, T_f, 0.98), ..., (T_b, T_p, 0.72), (T_b, T_q, 0.70)\}$. Finalmente, o modelo *SBERT* pôde ser treinado usando esses trios de treinamento, atribuindo-se os hiperparâmetros adicionais da Tabela 6.

Tabela 6 – Hiperparâmetros do *SBERT*

Hiperparâmetros	**Valores de referência**
embedding_size	$\{64, 128, 256, 768, 1024\}$
batch_size	32
evaluation_steps	100
epochs	$\{1, \ldots, 3\}$

Fonte: Elaborado pelo autor.

5.6 Avaliação da Eficácia das Abordagens para a Tarefa do Cálculo de Similaridade Espacial entre Trajetória

Considerando o exposto na subseção 4.1.2 que aborda o aumento dos dados de teste, constituídos pelo conjunto das trajetórias de pesquisa Q' e o conjunto das trajetórias de busca $S = Q'' \cup B''$. Para avaliar a eficácia das várias abordagens (modelos e métodos clássicos) analisadas neste trabalho quanto ao cálculo de similaridade espacial entre trajetórias, executaram-se os seguintes passos: para cada trajetória $T_{q'} \in Q'$, foram calculadas suas trajetórias espacialmente mais similares dentro de $S = Q'' \cup B''$. Idealmente, $T_{q''} \in S$ deve ser identificada como a trajetória mais similar à $T_{q'}$ e, assim, ranqueada em primeira posição como resposta da consulta. Assim, uma medida de classificação bem conhecida, ranque médio (*mr*), definido pela Equação 5.1, pode ser empregada para quantificar isso. Onde *rank* determina a posição da resposta correta para uma consulta $T_{q'} \in Q'$ no espaço de busca S ranqueado.

$$mr(Q', S) = \frac{1}{|Q'|} \sum_{T_{q'} \in Q'} rank(T_{q'}, S). \tag{5.1}$$

A implementação efetiva da função de *rank* depende da abordagem considerada. Nos métodos clássicos, são utilizadas funções de distância aplicadas diretamente aos pares de trajetórias entre Q' e S. Nesse contexto, a função *rank* ordena as trajetórias com base nas distâncias calculadas. Para as abordagens que geram incorporações de trajetórias, as trajetórias do conjunto de teste aumentado devem primeiro passar por uma discretização e, em seguida, são convertidas em *embeddings*. A função *rank* classifica as trajetórias com base nas distâncias euclidianas entre os *embeddings* das trajetórias em Q' e os *embeddings* daquelas em S.

Os tamanhos dos conjuntos $|Q'|$ e $|S|$ utilizados serão apresentados na avaliação experimental (seção 5.7).

5.7 Avaliação Experimental

Nesta seção, apresentam-se os experimentos realizados para avaliar as abordagens discutidas, segundo as questões de pesquisa introduzidas no Capítulo 1. Mais precisamente, a Subseção 5.7.1 visa responder à questão de pesquisa **RQ1** avaliando a qualidade dos *embeddings* obtidos pelos modelos de linguagem quando se considera conjunto de trajetórias densas. A Subseção 5.7.2 responde à **RQ1** considerando um conjunto de trajetórias esparsas. Já na Subseção 5.7.3, reponde-se à questão de pesquisa **RQ2**, mostrando o desempenho das várias abordagens concorrentes à medida que se aumenta o tamanho do espaço de busca. Finalmente, a Subseção 5.7.4 enfatiza as considerações finais com base nos resultados obtidos.

5.7.1 Respondendo RQ1: pesquisa da trajetória mais similar em um conjunto de trajetórias densas

Neste experimento, foi considerado o conjunto de dados Porto, trajetórias consideradas densas. Aqui, o conjunto de trajetórias de consulta Q' foi setado para 1.000 trajetórias e o tamanho do espaço de busca S foi gradualmente aumentado de 20.000 a 100.000 trajetórias. A avaliação das representações foi dada via métrica de ranque médio (*mr*), para saber quão bons eram os *embeddings* aprendidos na tarefa de calcular a similaridade especial entre trajetórias densas. Tal avaliação seguiu a metodologia descrita na seção 5.6. A Tabela 7 apresenta os resultados – a nomenclatura *BERT_X* indica o modelo *BERT* treinado que gerou *embeddings* de dimensão *X*.

Tabela 7 – Ranque médio (*mr*) em relação ao tamanho do espaço de busca $|\mathbf{S}|$, conjunto de dados Porto

	Porto				
$\|\mathbf{S}\|$	**20K**	**40K**	**60K**	**80K**	**100K**
DTW	18.28	26.29	35.27	45.53	54.27
EDR	23.52	43.04	71.01	91.09	120.04
LCSS	31.19	62.75	92.83	123.37	155.14
t2vec	2.44	3.68	5.08	6.84	8.30
Word2Vec	2.38	3.61	4.91	6.66	7.97
Doc2Vec	1.44	1.80	**2.16**	2.62	2.99
BERT_64	1.85	2.61	3.36	4.24	5.04
BERT_128	1.63	2.23	2.84	3.77	4.19
BERT_1024	**1.40**	**1.78**	2.17	**2.62**	**2.96**
SBERT	1.67	2.29	2.94	3.75	4.41

Fonte: Elaborado pelo autor.

A partir dos resultados, é possível notar que o desempenho dos modelos e técnicas cai à medida que o tamanho do espaço de busca *S* aumenta. Contudo, é possível observar que todos os modelos de linguagem conseguem superar os demais competidores, inclusive o *t2vec*. Isto possivelmente indica que os modelos de linguagem, especialmente os mais robustos como o *BERT*, têm o potencial de produzir consistentes representações (*embeddings*) de trajetórias com alta qualidade para a tarefa do cálculo de similaridade espacial entre trajetórias quando treinados em conjuntos de trajetórias densas. Por fim, observa-se que mesmo reduzindo gradualmente a dimensão dos *embeddings* para 64, o *BERT* ainda consegue superar significativamente o *t2vec*.

5.7.2 Respondendo RQ1: pesquisa da trajetória mais similar em um conjunto de trajetórias esparsas

Considerando os ótimos resultado mostrados na Subseção 5.7.1, no experimento corrente o objetivo foi avaliar se os mesmos modelos de linguagem podem alcançar um resultado parecido para um conjunto de trajetórias de perfil esparso. Para este fim, foi considerado o conjunto de trajetórias esparsas T-drive e aplicada a mesma avaliação metodológica. Devido ao baixo número de trajetórias no conjunto T-drive, o conjunto de trajetórias de consulta Q' foi setado para 500 trajetórias, e o tamanho do espaço de busca *S* variou de 10.000 a 50.000 trajetórias. A Tabela 8 apresenta os resultados obtidos – novamente, *BERT_X* indica o modelo *BERT* treinado com a dimensão *X*.

Tabela 8 – Ranque médio (*mr*) em relação ao tamanho do espaço de busca |**S**|, conjunto de dados T-drive

	T-drive				
|S|	**10K**	**20K**	**30K**	**40K**	**50K**
DTW	454.61	892.04	1327.98	1774.94	2214.63
EDR	587.55	1193.97	2089.50	2693.07	3472.31
LCSS	660.58	1355.62	2004.37	2677.75	3362.30
t2vec	**21.29**	**37.28**	**52.50**	**67.92**	**85.49**
Word2Vec	156.79	249.10	420.70	541.68	550.48
Doc2Vec	111.59	207.70	291.71	370.06	482.87
BERT_64	56.85	93.06	146.38	189.27	236.26
BERT_128	177.95	348.83	509.21	687.17	867.35
BERT_1024	94.95	162.23	280.19	310.17	402.55
SBERT	168.21	258.41	338.95	421.83	526.27

Fonte: Elaborado pelo autor.

A partir dos resultados, observa-se que os modelos de linguagem alcançam consistentemente resultados piores que o *t2vec*, sugerindo que estes são sensíveis a dados com

baixas taxas amostrais. Presumi-se que trajetórias esparsas podem ser análogas à noção de textos esparsos, que se refere a documentos ou conjuntos de dados de textuais com conteúdo limitado, ou informações esparsas, levando assim a menos informação contextual. Isto, por sua vez, pode explicar a qualidade inferior dos *embeddings* gerados por estes modelos em comparação com os do *t2vec*.

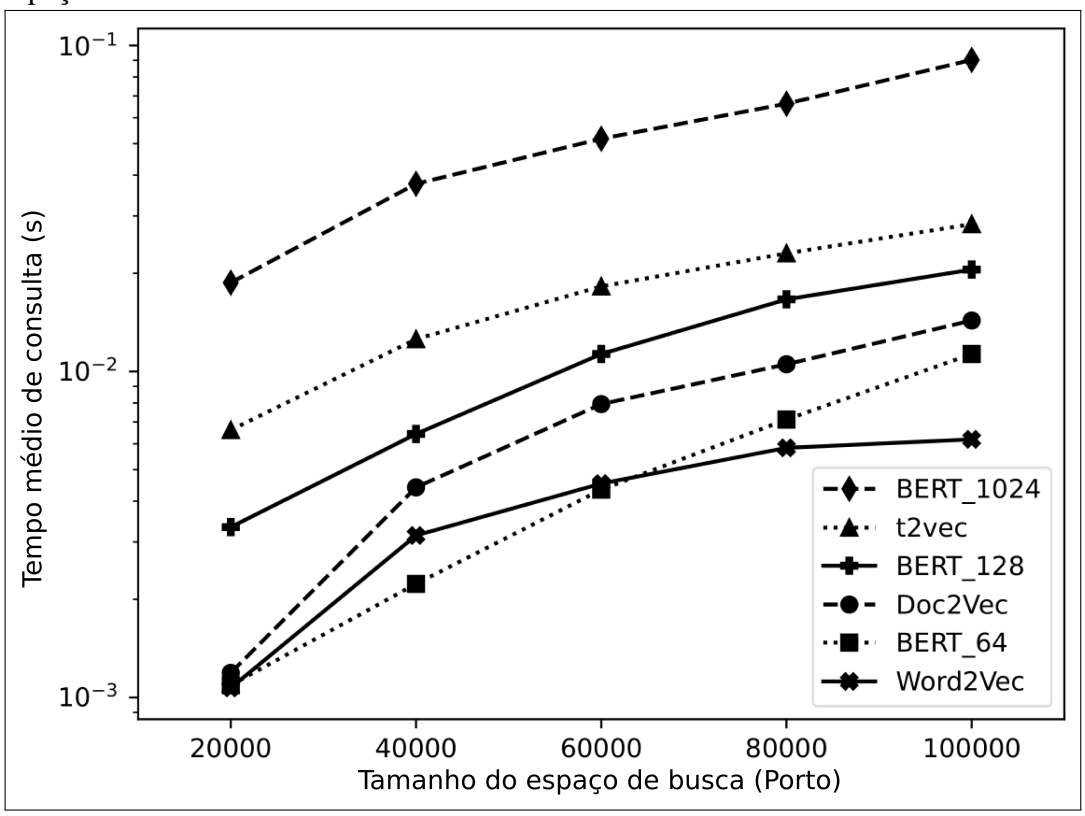
Uma analogia possível para explicar o bom desempenho do *t2vec* com dados esparsos é que ele utiliza uma função de perda (*loss function*) que incorpora a proximidade espacial durante o treinamento do modelo. Além disso, a representação das células, tokenização usada pelo *t2vec*, também leva em consideração a proximidade espacial das células aprendidas pelo modelo. Isso ressalta a capacidade do *t2vec* em aprender dados mais esparsos em comparação com os modelos de linguagem convencionais.

5.7.3 Respondendo RQ2: análise de escalabilidade

A escalabilidade desempenha um papel crítico na análise de similaridade espacial entre trajetórias. Os métodos clássicos, com sua complexidade quadrática em relação ao tamanho das trajetórias, tornam-se impraticáveis para grandes conjuntos de dados. Em contraste, recentes abordagens baseadas em aprendizado profundo, tais como o *t2vec*, alcançam complexidade linear com seus respectivos *embeddings* aprendidos. O experimento desta subseção visa avaliar a escalabilidade dos modelos de linguagem em relação ao *t2vec*. Para isso, foi usado o conjunto de dados Porto (o maior conjunto de dados avaliado neste trabalho) e considerado um conjunto de trajetórias de consultas Q' de tamanho 1.000. Já o espaço de busca S foi aumentado incrementalmente, variando seu tamanho de 20.000 a 100.000 trajetórias. Para cada combinação de Q' e S, todos os *embeddings* de suas trajetórias foram calculados e indexados usando uma árvore de busca (*kd-tree*). Esses passos foram conduzidos *offline* e, portanto, não foram incluídos nas medições de tempo. Em seguida, os k-vizinhos mais próximos (k-NN, com $k = 50$) foram calculados para cada trajetória de consulta incorporada em Q' em relação àquelas em S. Os resultados, mostrados na Figura 20, ilustram o tempo médio de cálculo para as 1.000 consultas.

Os resultados mostrados na Figura 20 confirmam que o tempo médio de consulta aumenta com o tamanho de S, conforme esperado. Contudo, é notável que o modelo *BERT* tem seu tempo médio de consulta variando com a dimensão do *embedding* (novamente, esperado). Porém, o mais interessante é que o *BERT* permanece competitivo em relação ao *t2vec*, apesar da sua arquitetura mais complexa. Considerando também os resultados mostrados na Seção 5.7.1,

Figura 20 – Análise de escalabilidade: calculando k-nn em relação ao tamanho do espaço de busca



Fonte: Elaborado pelo autor (2024).

isso sugere fortemente que o *BERT*, pelo menos em conjuntos de dados de trajetória densos, tem o potencial de produzir *embeddings* menores e de maior qualidade do que os concorrentes projetados especificamente para a tarefa de avaliação de similaridade espacial entre trajetórias, ao mesmo tempo que exibe forte escalabilidade.

5.7.4 Considerações Finais

Com base nas duas questões de pesquisa introduzidas no Capítulo 1, os resultados até aqui indicam que os modelos de linguagem são eficazes na geração de *embeddings* de alta qualidade para conjuntos de trajetórias densas, abrangendo assim parte das questões de pesquisa **RQ1** e **RQ2**. No entanto, este sucesso não se estendeu às trajetórias esparsas, como mostraram os resultados na subseção 5.7.2.

É possível supor, então, que trajetórias mais longas, as quais são mais comuns em conjuntos de dados densos como o Porto, contêm mais informação contextual e cobrem uma vasta gama de *tokens* (células na grade). Esta complexidade dos dados obriga os modelos a esforçarem-se para compreender as relações de longo prazo. Por exemplo, no conjunto de dados

Porto, o comprimento médio das trajetórias discretizadas é de cerca de 26 células da grade, podendo chegar a 100 em alguns casos. Os modelos de linguagem, particularmente o *BERT* com sua arquitetura *transformers*, são excelentes na captura dessas dependências em conjuntos de dados densos. Na verdade, os mecanismos de atenção do *BERT*, que ligam pares de *tokens* distantes, facilitam provavelmente a otimização e a aprendizagem de dependências de longo prazo (Vaswani *et al.*, 2017).

Por outro lado, trajetórias esparsas, como as do conjunto de dados T-Drive, apresentam lacunas maiores entre os pontos e transmitem menos detalhes de movimento, assemelhando-se à noção análoga de texto esparso com conteúdo limitado. O comprimento médio das trajetórias discretizadas no T-Drive é de cerca de 9 células da grade, e o tamanho do conjunto de dados é relativamente modesto, pouco mais de 500.000 trajetórias após o pré-processamento. Tais condições representam desafios para os grandes modelos linguísticos como o *BERT*, que prosperam em bases de dados extensas de treinamento. Da mesma forma, modelos como *Doc2Vec* e *Word2Vec*, que aprendem prevendo uma palavra-alvo (ou célula da grade) dentro de uma frase (ou trajetória), podem ter dificuldades com trajetórias curtas e esparsas, impactando seu desempenho. Além disso, os resultados confirmam que grandes modelos contextuais de linguagem, como o *BERT*, superam consistentemente as abordagens não contextuais, como o *Word2Vec*, ou modelos contextuais mais simples, como o *Doc2Vec*.

Finalmente, como visto na análise de escalabilidade direcionada à questão de pesquisa **RQ2** (subseção 5.7.3), o *BERT* demonstra forte escalabilidade, sugerindo seu potencial para gerar *embeddings* de trajetória de maior qualidade ao escolher a dimensão das incorporações adequadamente.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste estudo, foi explorada a tarefa de avaliar a similaridade espacial entre trajetórias aproveitando incorporações (*embeddings*) geradas a partir de modelos de linguagem pré-selecionados. Esses modelos foram reaproveitados para aprender os *embeddings* das trajetórias e demonstraram por meio da avaliação experimental que modelos de linguagem maiores, como o *BERT* e *SBERT*, podem produzir incorporações de alta qualidade para conjuntos de dados de trajetórias densas, mantendo a escalabilidade linear com o tamanho do espaço de busca. Infelizmente, esses mesmos modelos de linguagem não conseguiram sucesso semelhante quando treinados em conjuntos de dados esparsos. Embora resultados abaixo do ideal tenham sido previstos para os modelos mais simples, como *Word2Vec* e *Doc2Vec*, esperava-se um melhor desempenho para os modelos mais robustos (*BERT* e *SBERT*) quanto aos dados esparsos.

Outro ponto relevante deste estudo, relacionado à tarefa de avaliar a similaridade espacial entre trajetórias, é a capacidade dos *embeddings* de agirem como uma alternativa adicional para acelerar o cálculo de similaridade. O problema do cálculo de similaridade entre trajetórias possui complexidade quadrática $O(n^2)$, devido aos métodos clássicos que em sua maioria utilizam a programação dinâmica. No entanto, o uso de *embeddings* obtidos por meio de modelos de linguagem pode reduzir significativamente o tempo de execução para a ordem linear, $O(n+|v|)$, onde n representa o comprimento da trajetória e v a dimensão do seu respectivo *embedding*.

O uso de modelos de linguagem abre vários caminhos para pesquisas futuras. Uma direção potencial para melhoria dos resultados surge da observação de que os *LLMs* geralmente não são afetados por lacunas nas sequências, um cenário que o pipeline deste trabalhou tratou, eliminando lacunas (*gaps*) nas trajetórias causadas por células não quentes, ou seja, que não faziam parte do vocabulário. A modificação deste pipeline para manter as trajetórias em seu formato original poderia adaptá-lo melhor ainda às necessidades dos *LLMs*, melhorando potencialmente o seu desempenho em conjuntos de dados esparsos.

Além disso, em relação ao *LLM* específico para sentenças (*SBERT*), uma possível melhoria para trabalhos futuros poderia ser aplicada aos seus dados de treinamento. Em vez de mapear trajetórias semelhantes e menos semelhantes apenas variando a rota real R, uma abordagem mais viável poderia ser o uso combinado de alguma função de distância bem estabelecida que pudesse incorporar também as relações espaciais na formação das triplas de treinamento, acrescentando o conceito de trajetórias semelhantes e diferentes.

Outro caminho potencial para os trabalhos futuros envolve aprofundar-se em arquiteturas alternativas baseadas em *BERT*, adaptando a função de perda para melhor capturar as características espaciais das trajetórias (ou suas unidades discretizadas). Uma direção adicional de pesquisa também envolveria a exploração de *LLMs* ainda mais poderosos para a representação de trajetórias como, por exemplo, o *Generative Pre-trained Transformers* (GPT) e *Large Language Model Meta AI 2* (LlaMa2), conforme abordado em Gruver *et al.* (2023), onde séries temporais são codificadas como cadeias numéricas para previsão. Este conceito tem grande potencial de investigação, considerando-se as semelhanças inerentes entre trajetórias e séries temporais.

REFERÊNCIAS

ANDRADE, T.; GAMA, J. Identifying points of interest and similar individuals from raw gps data. In: CAGÁŇOVÁ, D.; HORŇÁKOVÁ, N. (Ed.). **Mobility Internet of Things 2018**. Cham: Springer International Publishing, 2020. p. 293–305. ISBN 978-3-030-30911-4.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. **IEEE Trans. Pattern Anal. Mach. Intell.**, IEEE Computer Society, USA, v. 35, n. 8, p. 1798–1828, aug 2013. ISSN 0162-8828. Disponível em: https://doi.org/10.1109/TPAMI.2013.50. Acesso em: 19 fev. 2022.

CAO, H.; XU, F.; SANKARANARAYANAN, J.; LI, Y.; SAMET, H. Habit2vec: Trajectory semantic embedding for living pattern recognition in population. **IEEE Transactions on Mobile Computing**, v. 19, n. 5, p. 1096–1108, 2020.

CHEN, L.; ÖZSU, M. T.; ORIA, V. Robust and fast similarity search for moving object trajectories. In: **Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data**. New York, NY, USA: Association for Computing Machinery, 2005. (SIGMOD '05), p. 491–502. ISBN 1595930604. Disponível em: https://doi.org/10.1145/1066157.1066213. Acesso em: 11 set. 2023.

CRIVELLARI, A.; RESCH, B.; SHI, Y. Tracebert – a feasibility study on reconstructing spatial–temporal gaps from incomplete motion trajectories via bert training process on discrete location sequences. **Sensors**, MDPI, v. 22, n. 4, p. 1682, 2022.

CRUZ, L.; SILVA, T. Coelho da; MAGALHÃES, R.; MELO, W.; CORDEIRO, M.; MACEDO, J. de; ZEITOUNI, K. Modeling trajectories obtained from external sensors for location prediction via nlp approaches. **Sensors**, v. 22, n. 19, 2022. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/22/19/7475. Acesso em: 02 jul. 2023.

CRUZ, L. A.; ZEITOUNI, K.; MACEDO, J. A. F. de. Trajectory prediction from a mass of sparse and missing external sensor data. In: **2019 20th IEEE International Conference on Mobile Data Management (MDM)**. Hong Kong, China: International Conference on Mobile Data Management, 2019. p. 310–319. ISBN 9781728133638.

CUDRE-MAUROUX, P.; WU, E.; MADDEN, S. Trajstore: An adaptive storage system for very large trajectory data sets. In: **2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)**. Long Beach, CA, USA: IEEE Computer Society, 2010. p. 109–120.

DAMIANI, M. L.; ACQUAVIVA, A.; HACHEM, F.; ROSSINI, M. Learning behavioral representations of human mobility. In: **Proceedings of the 28th International Conference on Advances in Geographic Information Systems**. New York, NY, USA: Association for Computing Machinery, 2020. (SIGSPATIAL '20), p. 367–376. ISBN 9781450380195. Disponível em: https://doi.org/10.1145/3397536.3422255. Acesso em: 17 dez. 2021.

DATA SCIENCE ACADEMY. **Deep Learning Book Brasil**. Disponível em: https://www.deeplearningbook.com.br/. Acesso em: 25 out. 2023.

DEVLIN, J.; CHANG, M.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In: **Proceedings of the 2019 Conference of the North American Chapter of the ACL: Human Language Technologies, Vol.1**. Minneapolis, Minnesota: ACL, 2019. p. 4171–4186. Disponível em: https://aclanthology.org/N19-1423. Acesso em: 23 jan. 2020.

FALOUTSOS, C.; RANGANATHAN, M.; MANOLOPOULOS, Y. Fast subsequence matching in time-series databases. **SIGMOD Rec.**, Association for Computing Machinery, New York, NY, USA, v. 23, n. 2, p. 419–429, may 1994. ISSN 0163-5808. Disponível em: https://doi.org/10.1145/191843.191925. Acesso em: 02 out. 2021.

FANG, Z.; DU, Y.; ZHU, X.; HU, D.; CHEN, L.; GAO, Y.; JENSEN, C. Spatio-temporal trajectory similarity learning in road networks. In: **Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2022. (KDD '22), p. 347–356. ISBN 9781450393850. Disponível em: https://doi.org/10.1145/3534678.3539375. Acesso em: 05 jan. 2024.

FU, T.-Y.; LEE, W.-C. Trembr: Exploring road networks for trajectory representation learning. **ACM Trans. Intell. Syst. Technol.**, Association for Computing Machinery, New York, NY, USA, v. 11, n. 1, feb 2020. ISSN 2157-6904. Disponível em: https://doi.org/10.1145/3361741. Acesso em: 14 ago. 2023.

GERON, A. **Mãos à obra**: aprendizado de máquina com scikit-learn e tensorflow. Rio de Janeiro - RJ: Alta Books, 2019.

GOMES, G. A. M. **Visualização interativa de dinâmicas de tráfego através de dados de trajetórias.** 127 f. Tese (Doutorado) – Universidade Federal do Ceará, Fortaleza, 2018.

GRUVER, N.; FINZI, M.; QIU, S.; WILSON, A. G. **Large Language Models Are Zero-Shot Time Series Forecasters**. Curran Associates, Inc., 2023. 19622–19635 p. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2023/file/3eb7ca52e8207697361b2c0fb3926511-Paper-Conference.pdf. Acesso em: 13 fev. 2024.

GUTING, R. H.; SCHNEIDER, M. Realm-based spatial data types: the rose algebra. **The International Journal on Very Large Data Bases (VLDBJ)**, v. 4, n. 2, p. 243–286, 1995.

HAYKIN, S. **Redes neurais**: princípios e prática. 2. ed. Porto Alegre - RS: Bookman, 2007.

HINTON, G. E.; MCCLELLAND, J. L.; RUMELHART, D. E. Distributed representations. In: **Parallel Distributed Processing: Explorations in the Microstructure of Cognition**. Cambridge, MA, USA: MIT Press, 1986. v. 1, p. 77–109. ISBN 026268053X.

KEOGH, E. J.; PAZZANI, M. J. Scaling up dynamic time warping for datamining applications. In: **Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2000. (KDD '00), p. 285–289. ISBN 1581132336. Disponível em: https://doi.org/10.1145/347090.347153. Acesso em: 16 mai. 2022.

KRUSKAL, J. B. An overview of sequence comparison: Time warps, string edits, and macromolecules. **Society for Industrial and Applied Mathematics (SIAM)**, Society for Industrial and Applied Mathematics, v. 25, n. 2, p. 201–237, 1983. ISSN 00361445. Disponível em: http://www.jstor.org/stable/2030214. Acesso em: 03 jun. 2023.

LE, Q.; MIKOLOV, T. Distributed representations of sentences and documents. In: **Proceedings of the 31st International Conference on Machine Learning**. Bejing, China: PMLR, 2014. (Proceedings of Machine Learning Research, 2), p. 1188–1196. Disponível em: https://proceedings.mlr.press/v32/le14.html. Acesso em: 12 set. 2021.

LEE, J.-G.; HAN, J.; WHANG, K.-Y. Trajectory clustering: a partition-and-group framework. In: **Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data**. New York, NY, USA: Association for Computing Machinery, 2007. (SIGMOD '07), p. 593–604. ISBN 9781595936868. Disponível em: https://doi.org/10.1145/1247480.1247546. Acesso em: 07 jul. 2022.

LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions, and reversals. **Soviet physics. Doklady**, Cybernetics and Control Theory, v. 10, n. 8, p. 707–710, 1965.

LI, H.; LIU, J.; WU, K.; YANG, Z.; LIU, W.; XIONG, N. Spatio-temporal vessel trajectory clustering based on data mapping and density. **IEEE Access**, v. 6, p. 58939–58954, 08 2018.

LI, X.; ZHAO, K.; CONG, G.; JENSEN, C. S.; WEI, W. Deep representation learning for trajectory similarity computation. In: **34th IEEE International Conference on Data Engineering (ICDE)**. Paris, France: IEEE, 2018. p. 617–628.

LI, Z.; DING, B.; HAN, J.; KAYS, R. Swarm: mining relaxed temporal moving object clusters. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 3, n. 1–2, p. 723–734, sep 2010. ISSN 2150-8097. Disponível em: https://doi.org/10.14778/1920841.1920934. Acesso em: 11 set. 2023.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **Proceedings of Workshop at ICLR**, v. 2013, 01 2013. Disponível em: https://dblp.org/rec/journals/corr/abs-1301-3781. Acesso em: 28 abr. 2022.

PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. Deep contextualized word representations. In: WALKER, M.; JI, H.; STENT, A. (Ed.). **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 2227–2237. Disponível em: https://aclanthology.org/N18-1202. Acesso em: 21 ago. 2023.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. Improving language understanding with unsupervised learning. **Technical Report - OpenAI**, OpenAI Research, 2018.

RANU, S.; P, D.; TELANG, A. D.; DESHPANDE, P.; RAGHAVAN, S. Indexing and matching trajectories under inconsistent sampling rates. In: **IEEE 31st International Conference on Data Engineering**. Seoul, Korea (South): International Conference on Data Engineering, 2015. p. 999–1010.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning - Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2**. 3. ed. Birmingham, UK: Packt Publishing, 2019.

REIMERS, N.; GUREVYCH, I. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**. Hong Kong, China: Association for Computational Linguistics, 2019. p. 3982–3992. Disponível em: https://aclanthology.org/D19-1410. Acesso em: 12 jan. 2022.

ROUCOS, S.; DUNHAM, M. A stochastic segment model for phoneme-based continuous speech recognition. In: **IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)**. Dallas, TX, USA: IEEE, 1987. v. 12, p. 73–76.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by back-propagating errors. **Nature**, Nature 323, p. 533–536, 1986. Disponível em: https://doi.org/10.1038/323533a0. Acesso em: 29 mai. 2022.

SALVADOR, S.; CHAN, P. Toward accurate dynamic time warping in linear time and space. **Intell. Data Anal.**, IOS Press, NLD, v. 11, n. 5, p. 561–580, oct 2007. ISSN 1088-467X.

SANDERSON, A. C.; WONG, A. K. C. Pattern trajectory analysis of nonstationary multivariate data. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 10, n. 7, p. 384–392, 1980.

SHUNCHENG, L.; SU, H.; ZHENG, B.; ZHOU, X.; ZHENG, K. A survey of trajectory distance measures and performance evaluation. **The VLDB Journal 29**, Spring-Verlag GmbH Germany, v. 408, n. 0949, p. 3–32, 2019. Disponível em: https://doi.org/10.1007/s00778-019-00574-9. Acesso em: 14 jul. 2022.

SU, H.; ZHENG, K.; WANG, H.; HUANG, J.; ZHOU, X. Calibrating trajectory data for similarity-based analysis. In: **Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data**. New York, NY, USA: Association for Computing Machinery, 2013. (SIGMOD '13), p. 833–844. ISBN 9781450320375. Disponível em: https://doi.org/10.1145/2463676.2465303. Acesso em: 05 mar. 2022.

TAGHIZADEH, S.; ELEKES, A.; SCHäLER, M.; BöHM, K. How meaningful are similarities in deep trajectory representations? **Information Systems**, Information Systems - Elsevier Ltd., v. 98, p. 101452, 2021. ISSN 0306-4379. Disponível em: https://www.sciencedirect.com/science/article/pii/S0306437919305046. Acesso em: 21 jan. 2024.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. In: **Proceedings of the 31st International Conference on Neural Information Processing Systems**. Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 6000–6010. ISBN 9781510860964.

VLACHOS, M.; KOLLIOS, G.; GUNOPULOS, D. Discovering similar multidimensional trajectories. In: **Proceedings 18th International Conference on Data Engineering**. San Jose, CA, USA: IEEE, 2002. p. 673–684.

WANG, H.; SU, H.; ZHENG, K.; SADIQ, S.; ZHOU, X. An effectiveness study on trajectory similarity measures. In: **Proceedings of the Twenty-Fourth Australasian Database Conference**. AUS: Australian Computer Society, Inc., 2013. (ADC '13, v. 137), p. 13–22. ISBN 9781921770227.

WANG, S.; CAO, J.; PHILIP, S. Y. Deep learning for spatio-temporal data mining: A survey. **IEEE transactions on knowledge and data engineering**, IEEE, v. 34, n. 8, p. 3681–3700, 2020. Disponível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9204396. Acesso em: 22 dez. 2022.

YANG, P.; WANG, H.; ZHANG, Y.; QIN, L.; ZHANG, W.; LIN, X. T3s: Effective representation learning for trajectory similarity computation. In: **IEEE 37th International Conference on Data Engineering (ICDE)**. Chania, Greece: IEEE, 2021. p. 2183–2188.

YAO, D.; CONG, G.; ZHANG, C.; BI, J. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In: **35th IEEE International Conference on Data Engineering (ICDE)**. Macau SAR, China: IEEE, 2019. p. 1358–1369. ISBN 9781728191843.

ZHANG, H.; ZHANG, X.; JIANG, Q.; ZHENG, B.; SUN, Z.; SUN, W.; WANG, C. Trajectory similarity learning with auxiliary supervision and optimal matching. In: **Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence**. Yokohama, Japan: IJCAI'20, 2021. ISBN 9780999241165.

ZHANG, Y.; LIU, A.; LIU, G.; LI, Z.; LI, Q. Deep representation learning of activity trajectory similarity computation. In: **2019 IEEE International Conference on Web Services (ICWS)**. Milan, Italy: IEEE, 2019. p. 312–319. ISBN 9781728127170.

ZHAO, W.; ZHOU, N.; SUN, A.; WEN, J.-R.; HAN, J.; CHANG, E. A time-aware trajectory embedding model for next-location recommendation. **Knowledge and Information Systems**, v. 56, 09 2018. Disponível em: https://doi.org/10.1007/s10115-017-1107-4. Acesso em: 01 mar. 2023.

ZHENG, Y. Trajectory data mining: An overview. **ACM Trans. Intell. Syst. Technol.**, Association for Computing Machinery, New York, NY, USA, v. 6, n. 3, may 2015. ISSN 2157-6904. Disponível em: https://doi.org/10.1145/2743025. Acesso em: 27 set. 2023.

ZHUANG, Z.; KONG, X.; ELKE, R.; ZOUAOUI, J.; ARORA, A. Attributed sequence embedding. In: **IEEE International Conference on Big Data (Big Data)**. Los Angeles, CA, USA: BIG-DATA, 2019. p. 1723–1728. Disponível em: https://doi.ieeecomputersociety.org/10.1109/BigData47090.2019.9006481. Acesso em: 19 abr. 2022.