Spatio-Temporal Data Mining --- Trajectory Mining and Applications

Gao Cong

Nanyang Technological University

Smart Nation Applications

GeoSpatial Data Mining

- POI recommenda tion & prediction
- Interactive exploration geospatial data
- Knowledge graph for locations
- Trajectory representation and similarity
- Spatial temporal prediction: Speed, travel time, route prediction
- Region search, (e.g., burst region)
- Region exploration (topic, crowdness)

Querying and indexing spatio-temporal data

- Snapshot queries (OLTP, OLAP)
- Continuous queries

Distributed streaming systems

- Distributed load balance
  Distributed materialized view
- Index &query optimizer
- Machine learning techniques

Big static/streaming geo-spatial + X (e.g., text, temporal) data

Trajectory Data

Player trajectory	Vehicle trajectory	Pedestrian trajectory	Animal trajectory	...

Outline

- Trajectory Level geospatial data mining
  - Trajectory representation and similarity (ICDE'18, ICDE'19, KDD'19)
    - Trajectory
    - A group of trajectories
  - Sub-trajectory similarity (VLDB'20)
  - Trajectory data and its applications in intelligent transport
    - Travel speed estimation (WWW'19)
    - Travel route estimation (ICDE'20)
  - Crowdness estimation

references

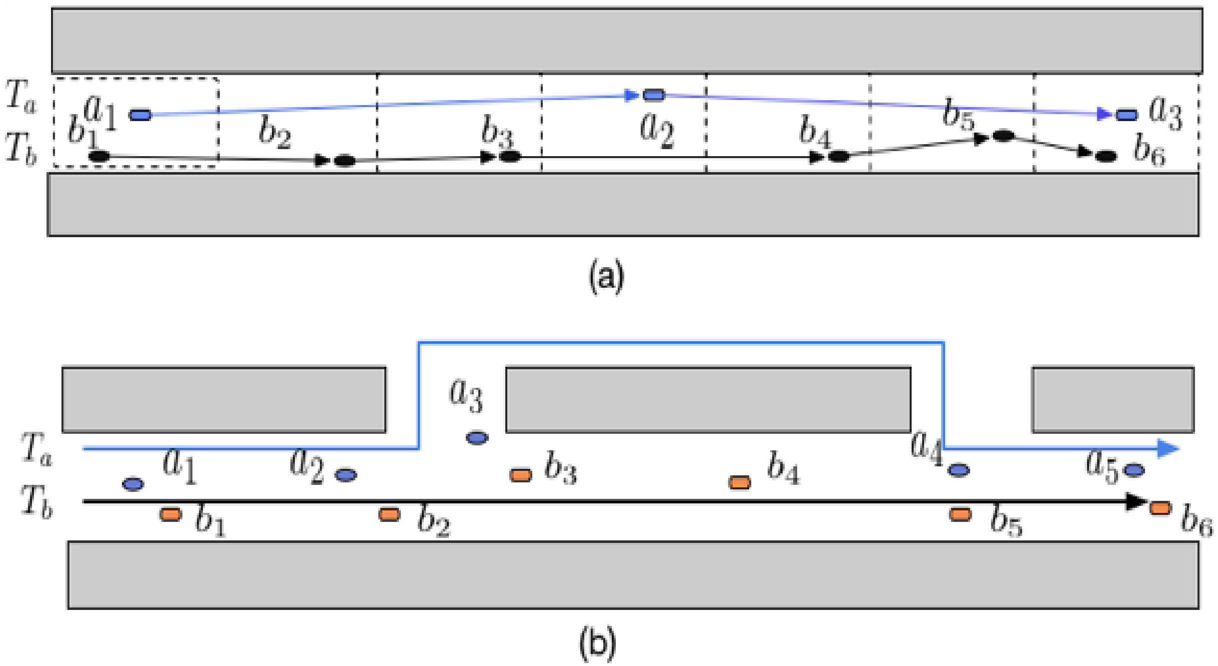
- Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, Wei Wei: Deep Representation Learning for Trajectory Similarity Computation. ICDE 2018: 617-628
- Zheng Wang, Cheng Long, Gao Cong, Ce Ju: Effective and Efficient Sports Play Retrieval with Deep Representation Learning. KDD 2019: 499-509
- Zheng Wang, Cheng Long, Gao Cong, Yiding Liu: Efficient and Effective Similar Subtrajectory Search with Deep Reinforcement Learning. Proc. VLDB Endow. 13(11): 2312-2325 (2020)
- Xiucheng Li, Gao Cong, Aixin Sun, Yun Cheng: Learning Travel Time Distributions with Deep Generative Model. WWW 2019: 1017-1027
- Xiucheng Li, Gao Cong, Yun Cheng: Spatial Transition Learning on Road Networks with Deep Probabilistic Models. ICDE 2020: 349-360
- Ali Zonoozi, Jung-jae Kim, Xiao-Li Li, Gao Cong: Periodic-CRN: A Convolutional Recurrent Model for Crowd Density Prediction with Recurring Periodic Patterns. IJCAI 2018: 3732-3738

Motivations

- The traditional methods measure the trajectory similarity by forming pairwise matching the sample points and identifying the best alignment, such as:
  - Dynamic Time Wrappings (DTW)
  - Longest Common Subsequence (LCSS)
  - Edit Distance on Real sequences (EDR)
  - Hausdorff and Frechet distance

- Drawbacks of these methods :
  - The non-uniform sampling rates
  - The low sampling rates
  - The noisy sample points
  - Quadratic computational cost (Dynamic Programming)

Illustration examples



a) Non-uniform sampling rates lead to undesired matching
b) Low-sampling rates make it hard to distinguish distinct underlying routes

Opportunities

- The problem with the two examples: the raw trajectory representation does not explicitly reveal its true route
  - Transition patterns are hidden in the real world trajectory data set
- Recent success in representation learning motivates us to learn trajectory representation that could encode its underlying route information

Challenges

Natural to consider the use of Recurrent Neural Nets (RNN) for its representation learning. However,

- The representation obtained using RNNs is still unable to reveal the most likely route
  - we consider a seq2seq based model to maximize the probability of recovering the true route of trajectory
- The existing loss functions used to train RNNs fail to consider the spatial proximity inherent in spatial data
  - we design a spatial proximity aware loss and pre-training algorithm

Problem statement

learn representation V for the trajectory T such that the representation can reveal the true underlying route of the trajectory in the similarity computation.

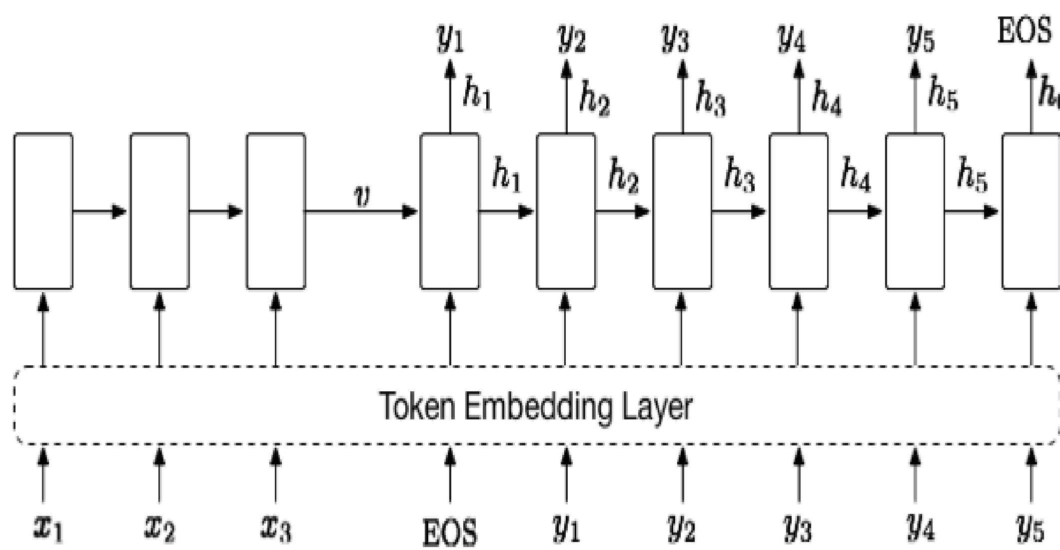
- robust to non-uniform, low sampling rates, noisy sample points
- scale to large-scale data sets.

The framework of the proposed method

to maximize the probability of the true route R conditioning on the observed trajectory T, i.e., $\mathbb{P}(R|T)$

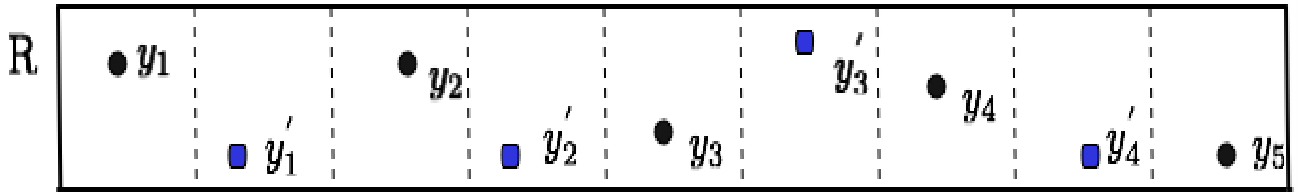
- Compressing the variable-length T into a vector $\mathbf{v}$
- Recovering the true route R from $\mathbf{v}$

$\mathbb{P}(R|T)$ is implemented as



the proposed method

- Partition the spatial space into the equal size cells and transform a trajectory into a sequence of discrete tokens:



$$T_1 = [y_1, \ldots, y_5]$$

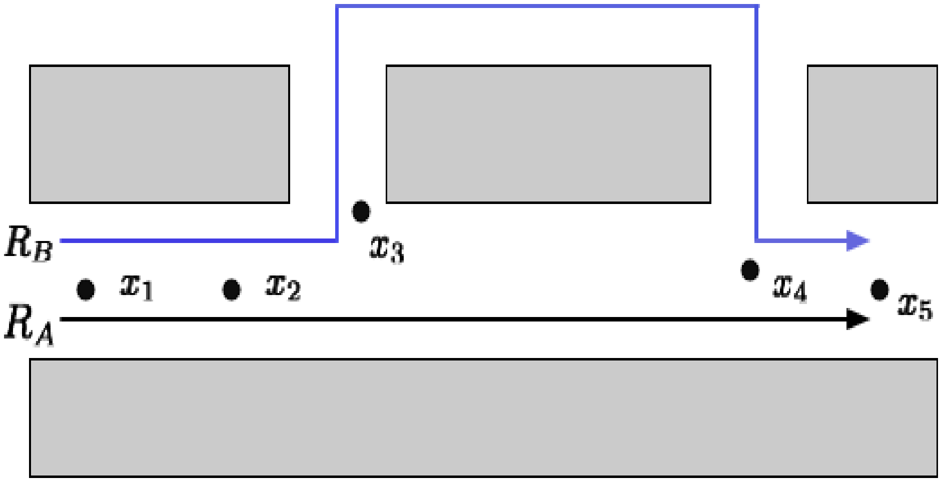
$$T_1' = [y_1', \ldots, y_4']$$

The proposed method

The model learns the hidden transition patterns. If there are multiple candidate routes R that could generate T, the model will recover the most likely one, and encode it into $\mathtt{V}$.

if R_B is popular than R_A, the model will learn

$$\mathbb{P}(R_B|T) > \mathbb{P}(R_A|T)$$



Challenges

- The underlying route R is not available
- The most adopted Negative Log Likelihood (NLL) loss might not guide the model learn the consistent representations for trajectories generated from the same route.

Proposed solutions

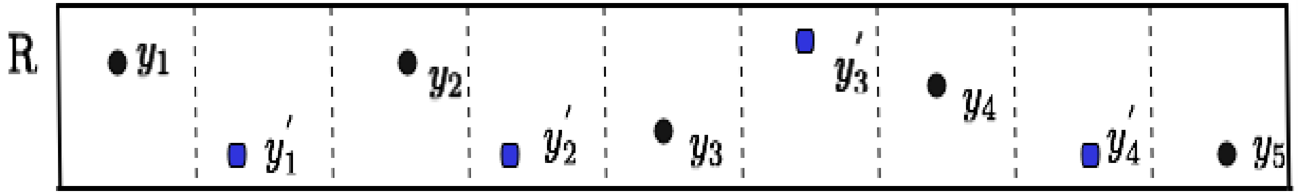
To address the first challenge, we exploit two observations:

- Both a non-uniform, low sampling rate trajectory T_a and a high sampling rate trajectory T_b are the paraphrases of their underlying route
- The high sampling rate one is closer to its true underlying route than the low sampling rate one.

$$\text{maximize} \quad \mathbb{P}(R|T_a) \Rightarrow \text{maximize} \quad \mathbb{P}(T_b|T_a) \qquad .$$

Proposed solutions

NLL differentiates $T_b = \{y_i\}, T_{b'} = \{y_i'\}$ as two trajectories.



NLL penalizes the output cells with equal weight.

$$\mathcal{L}_1 = -\log \prod_t \mathbb{P}\left(y_t | y_{1:t-1}, x\right)$$

Intuitively, the output cells that are closer to the target cell should be more acceptable than those that are far away.

Spatial proximity aware loss

spatial proximity loss function:

$$\mathcal{L}_2 = -\sum_{t=1}^{|y|} \sum_{u \in V} w_{uy_t} \log \frac{\exp\left(W_u^T h_t\right)}{\sum_{v \in V} \exp\left(W_v^T h_t\right)}$$

$$w_{uy_t} = \frac{\exp\left(-\|u - y_t\|_2 / \theta\right)}{\sum_{v \in V} \exp\left(-\|v - y_t\|_2 / \theta\right)}$$

consistent representations for trajectories generated from the same route.

Complexity of similarity computation

The proposed model can be trained unsupervised with SGD based algorithm.

Given a trained model, it takes $O(n)$ to encode a trajectory to vector V and then we can use Euclidean distance to compute two vectors similarity, with time complexity $O(|v|)$. Overall, the time cost of similarity computation is

$$O(n + |\mathbf{v}|)$$

Experimental setup

Dataset	#Points	#Trips	Mean length
Porto	74,269,739	1,233,766	60
Harbin	184,809,109	1,527,348	121

- The first 0.8 million trajectories is used for training, remaining for testing
- Two kinds of transformation of T_b
  - Dropping rate $r_1 = [0, 0.2, 0.4, 0.6]$
  - Distorting rate $r_2 = [0, 0.2, 0.4, 0.6]$

Benchmarking methods

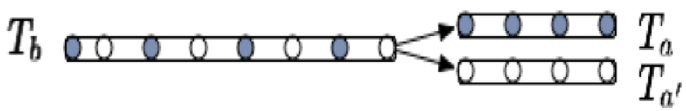
- EDR (Edit Distance on Real sequences)
- LCSS (Longest Common Subsequence)
- EDwP (Edit Distance with Projection)
- vRNN (vanilla RNN)
- CMS (Common Set representation)

- Both our code and dataset are publicly available.

Experiments: Most similar search

The lack of ground-truth makes it challenging to evaluate the accuracy.

Q	P
10k	m



$D_Q = \{T_a\}$	$D'_Q = \{T_{a'}\}$	D_P	D'_P
10k	10k	m	m

- Randomly select Q and P from test data set $Q \cap P = \emptyset$
- For each query $T_a \in D_Q$, we retrieve its top-k most similar ones from $D'_Q \cup D'_P$
- Calculate the mean rank of $T_{a'}$

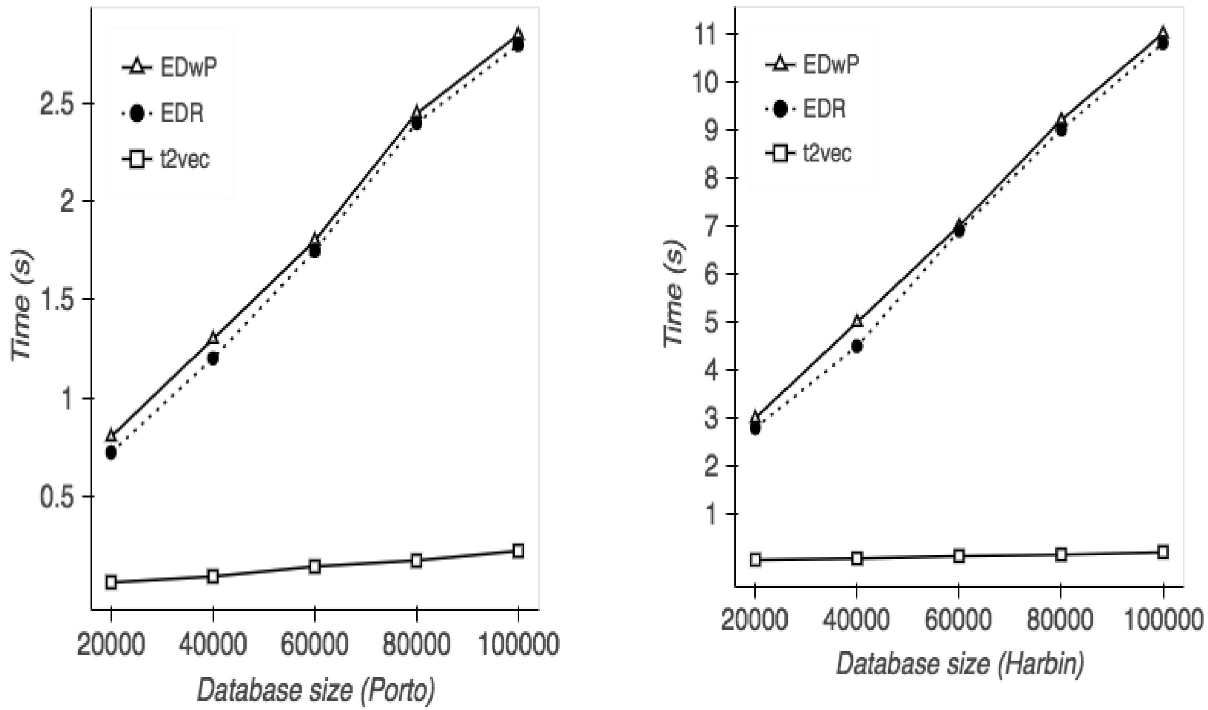
Most similar search

Increasing the size of database P.

Porto					
DB size	20k	40k	60k	80k	100k
EDR	25.73	50.70	76.07	104.01	130.98
LCSS	31.95	59.20	95.85	130.40	150.67
CMS	62.18	112.84	173.34	231.55	291.26
vRNN	32.73	61.24	100.20	135.22	163.10
EDwP	6.78	11.48	16.08	23.02	28.90
t2vec	**2.30**	**3.45**	**4.73**	**6.35**	**7.67**

Harbin					
DB size	20k	40k	60k	80k	100k
EDR	30.37	57.90	85.72	118.02	149.01
LCSS	35.49	63.20	105.46	137.20	160.67
CMS	97.41	141.04	209.37	271.45	316.81
vRNN	34.30	65.24	103.05	140.25	162.10
EDwP	12.80	20.64	29.10	35.20	45.30
t2vec	**5.10**	**7.50**	**9.62**	**12.51**	**15.70**

Scalability

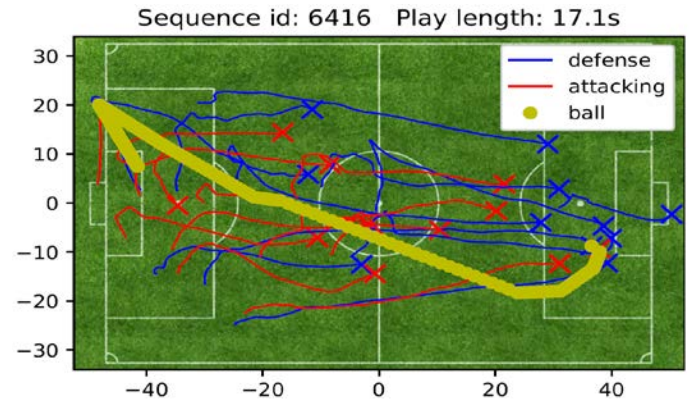


- EDR and EDwP employ carefully designed pruning and indexing techniques.
- t2vec is at least one order of magnitude faster
- t2vec supports interactive use and analysis on big trajectory

Outline

- Trajectory Level geospatial data mining
  - Trajectory representation and similarity (ICDE’18, ICDE’19, KDD’19)
    - Trajectory
    - A group of trajectories
  - Sub-trajectory similarity (VLDB’20)
  - trajectory data and its applications in intelligent transport
    - Travel speed estimation (WWW’19)
    - Travel route estimation (ICDE’20)
  - Crowdness estimation

Sports Data

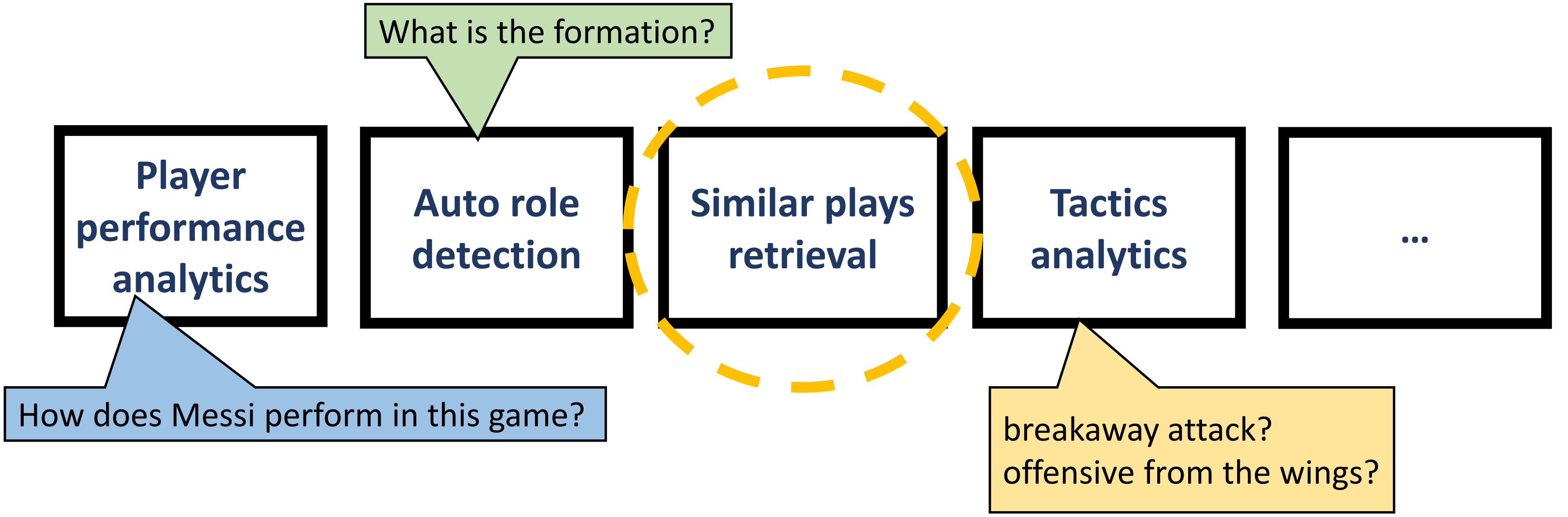


How far has Messi run during a game?

Visual sports data:
- Widely used and accessed
- Does not support quantitative analytics well

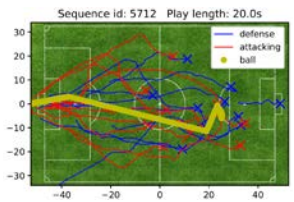
Spatiotemporal sports data:
- New perspectives for sports analytics
- Quantitative analytics

Spatiotemporal Sports Data Analytics

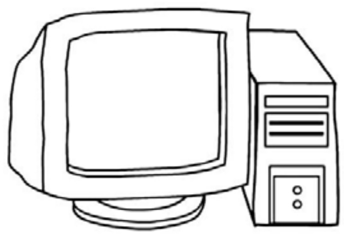


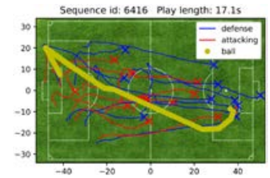
Similar Plays Retrieval

Similar Play Retrieval:
Find k plays that are the most **similar** to the query play

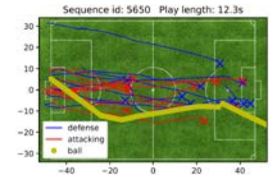


Query play

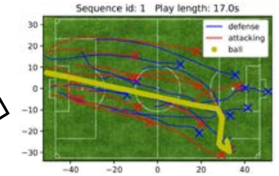




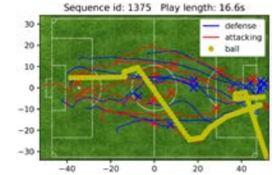
Play 1



Play 2



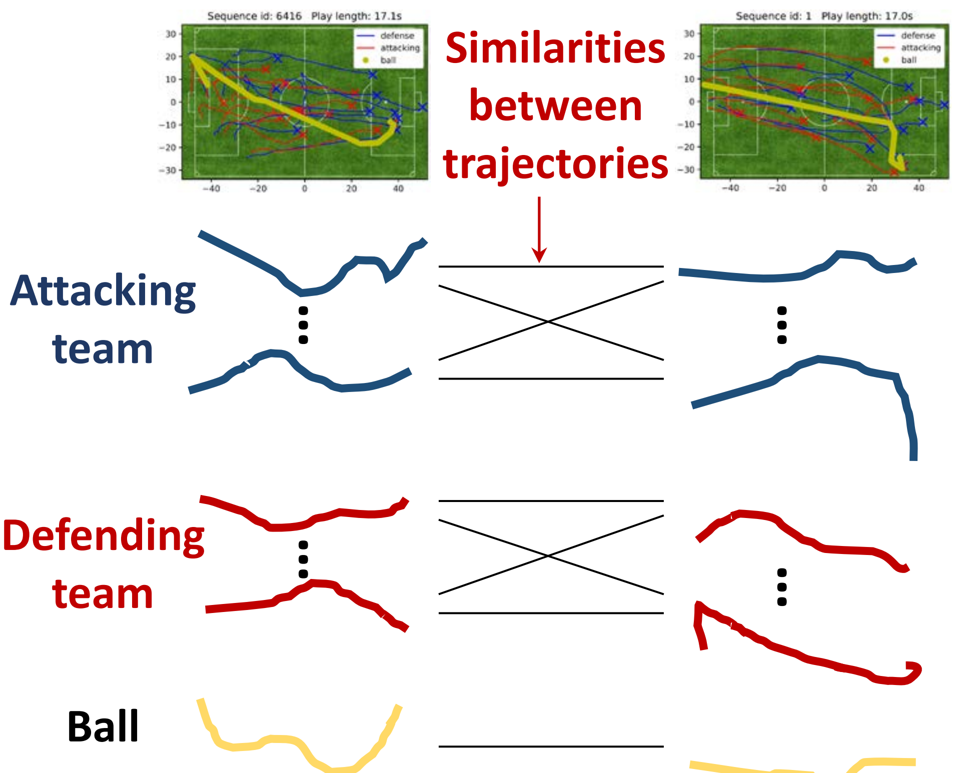
Play K-1



Play K

Key operator:
Measure the **similarity** between two plays

Existing Play Similarity Measurements (1) – Matching-based Alignment Approach



Efficiency Issue:
- **Quadratic** for computing the weight of each edge
- **Cubic** for computing an optimal alignment

Effectiveness Issue:
- A (f1) is not similar to A (f2), B(f1) and B(f2)
- figure1 is similar to figure2
- noise

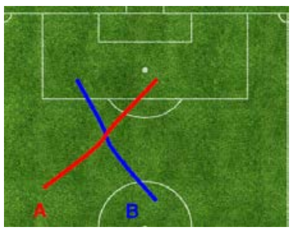


figure1

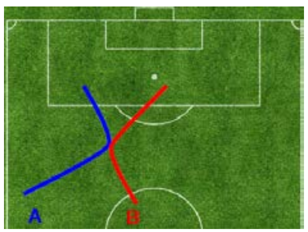
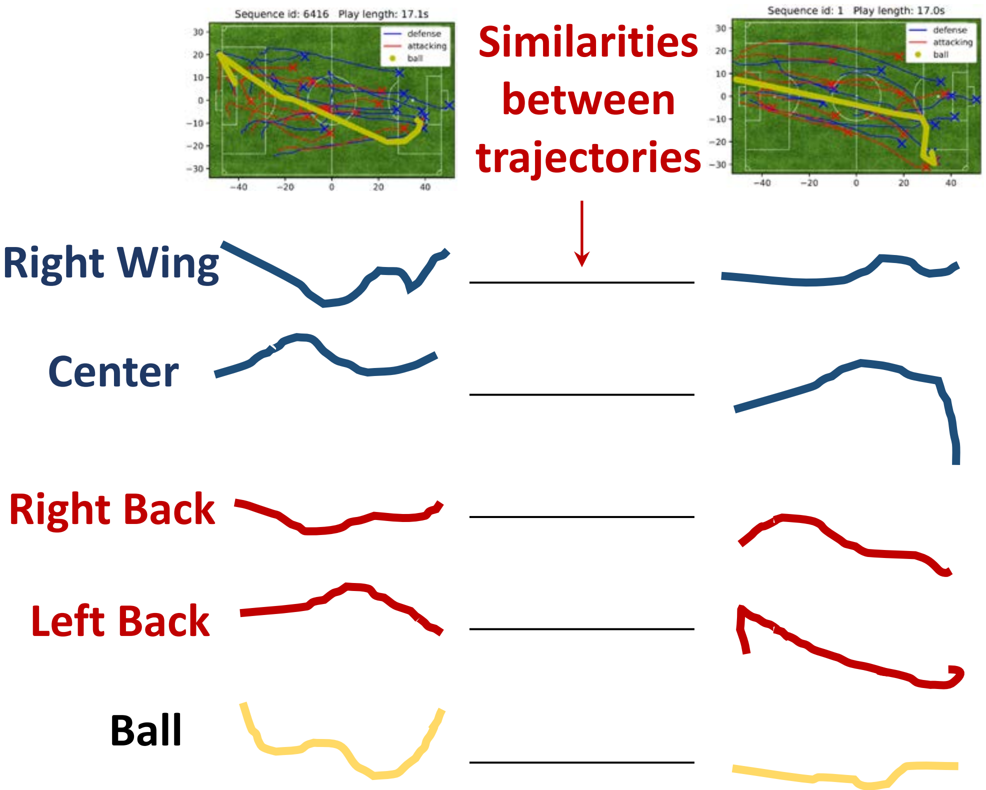


figure2

Existing Play Similarity Measurements (2) – Role-based Alignment Approach

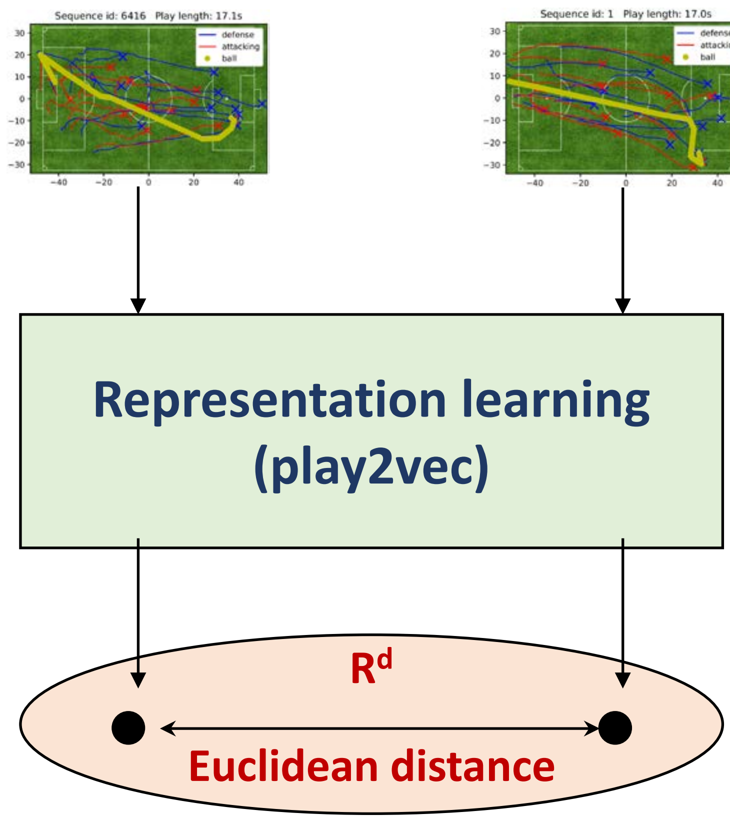


Efficiency Issue:
- **Quadratic** for computing the weight of each edge

Effectiveness Issue:
- **Assumptions** not justified

Robustness Issue:
- No mechanisms for handling **noises**

New Play Similarity Measurement – A Representation Learning Approach (play2vec)

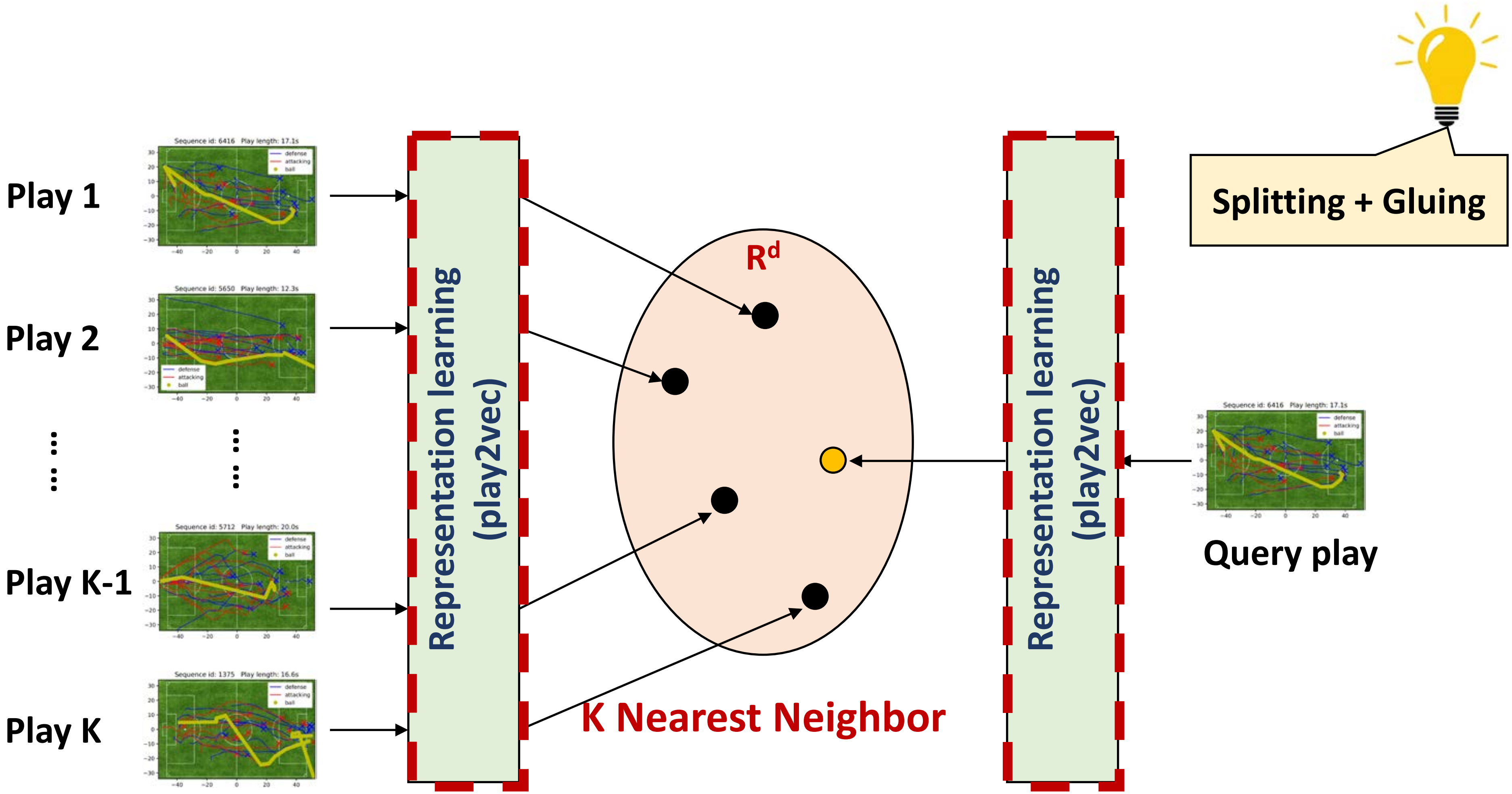


Efficiency advantage:
- **Linear** time complexity

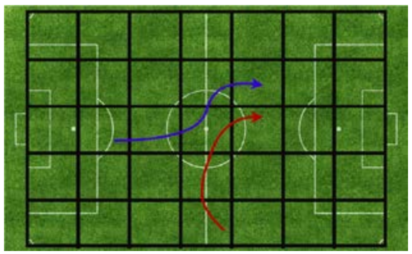
Effectiveness advantage:
- **Data-driven**

Robustness advantage:
- **De-noising** mechanism

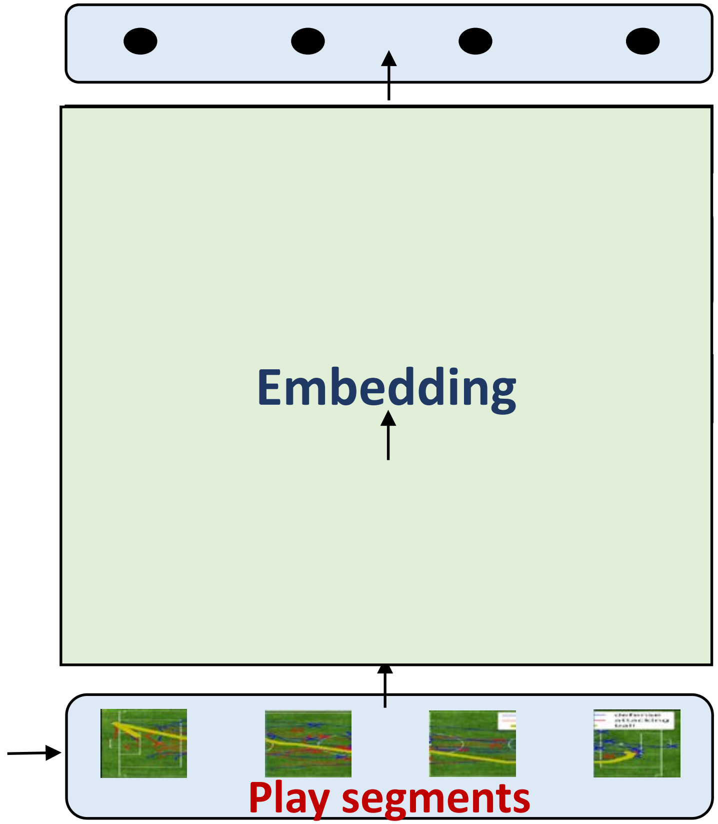
Similar Plays Retrieval based on Representation Learning (play2vec)

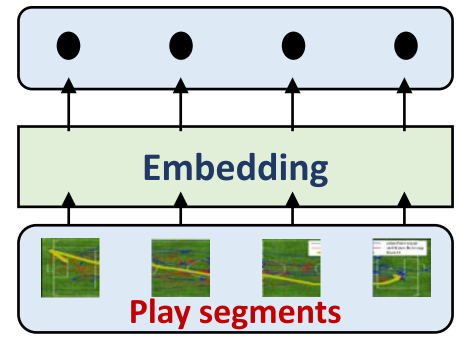


play2vec: splitting

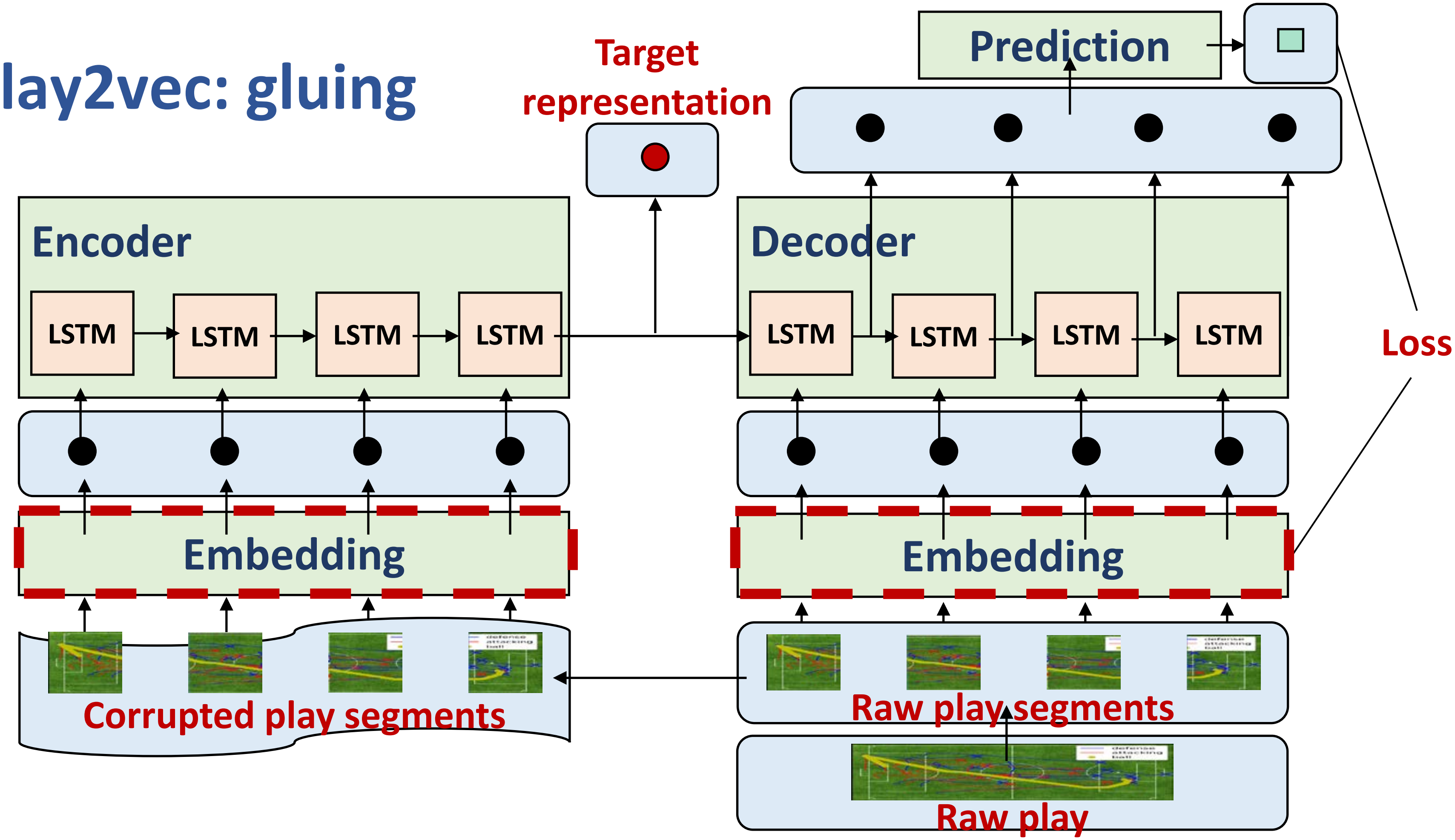


$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$





play2vec: gluing



Experimental Set-up

STATS

Statistics	Frequency
#Sequences	7500
Playing Time	45 games
Data Points	30.4M
X-axis	$[-52.5meters, +52.5meters]$
Y-axis	$[-34meters, +34meters]$
Sampling Rate	10Hz

Algorithms
DTW
Frechet
Chalkboard
EMDT
play2vec

Experiments	Performance metrics
Effectiveness	**Self-similarity**
	Cross-similarity
	KNN-similarity
Efficiency	**Running time**
Usability	**Case studies**

Effectiveness Experimental Results (1) – Self-similarity

Table 3: Self-similarity of varying noise rate.

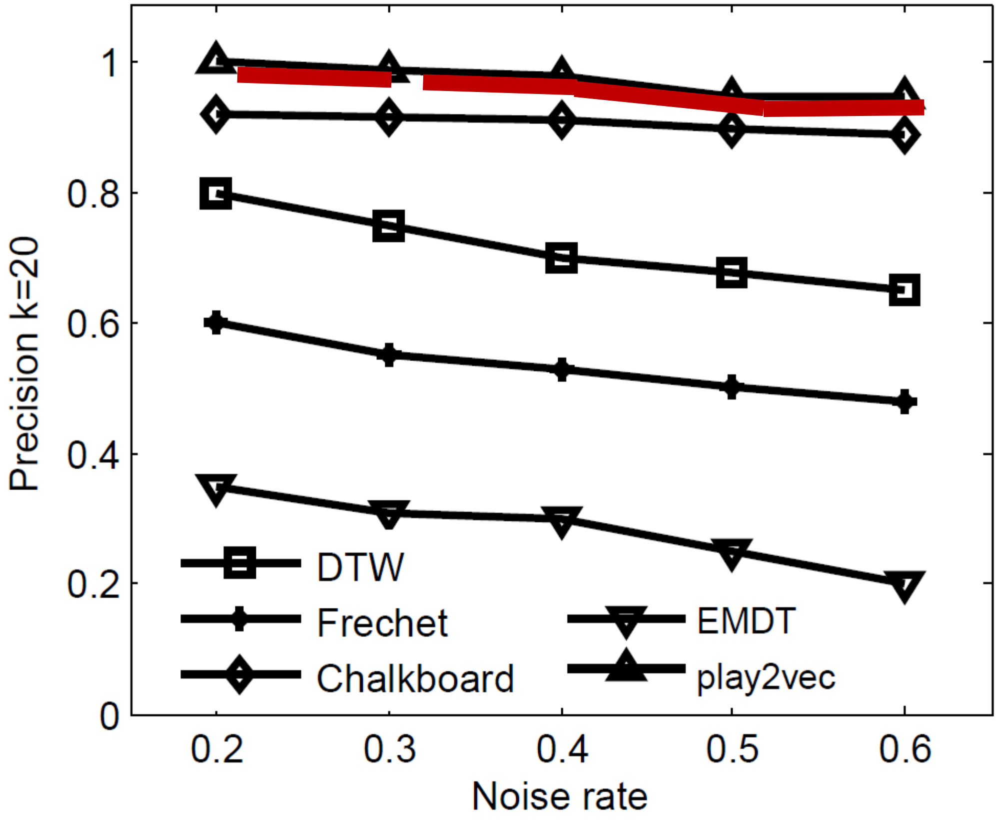
noise rate	0.2	0.3	0.4	0.5	0.6
DTW	24.80	33.80	44.00	73.20	90.20
Frechet	79.40	80.60	82.80	83.00	83.60
Chalkboard	77.20	77.80	78.20	78.40	78.80
EMDT	215.00	220.80	236.20	255.20	299.40
play2vec	**14.20**	**20.40**	**23.80**	**25.30**	**28.20**

Effectiveness Experimental Results (2) – Cross-similarity

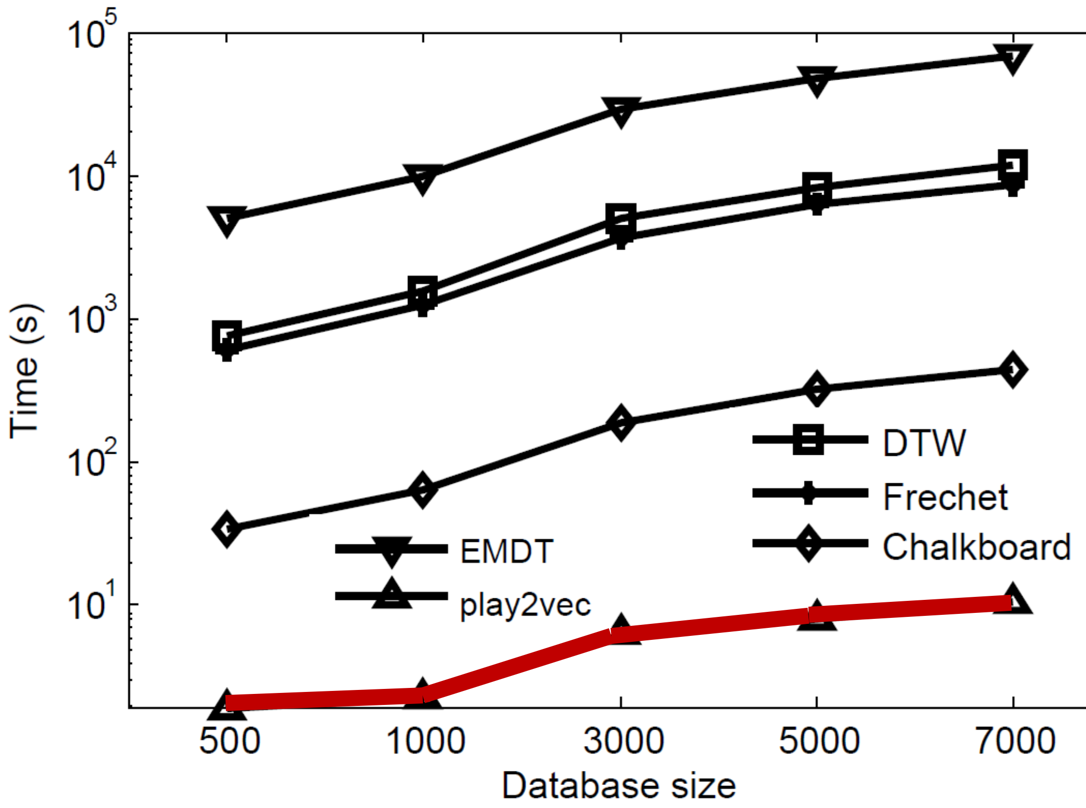
Table 5: Cross-similarity of varying noise rate.

noise rate	0.2	0.3	0.4	0.5	0.6
DTW	0.093	0.111	0.113	0.114	0.117
Frechet	0.084	0.101	0.102	0.105	0.116
Chalkboard	0.074	0.082	0.088	**0.093**	0.112
EMDT	0.583	0.629	0.706	0.819	0.891
play2vec	**0.034**	**0.048**	**0.086**	**0.093**	**0.111**

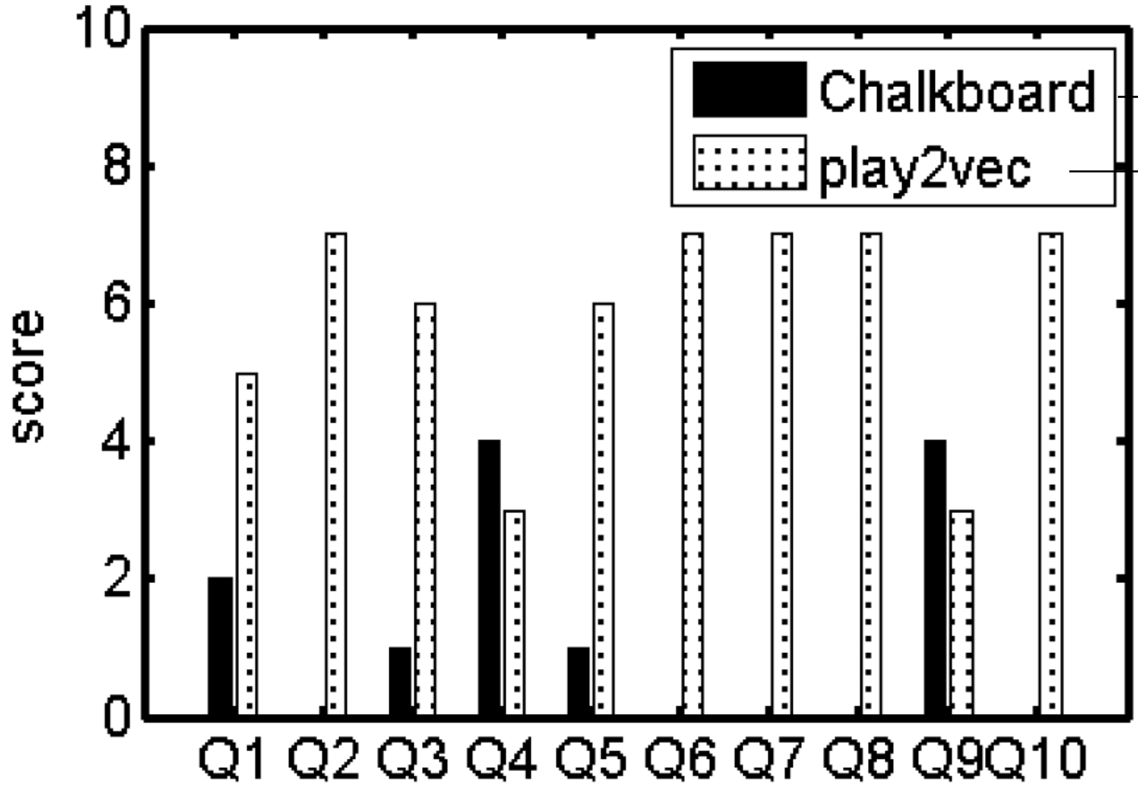
Effectiveness Experimental Results (3) – KNN-similarity



Efficiency Experimental Results – Running Time



Usability Experimental Results – Case Studies



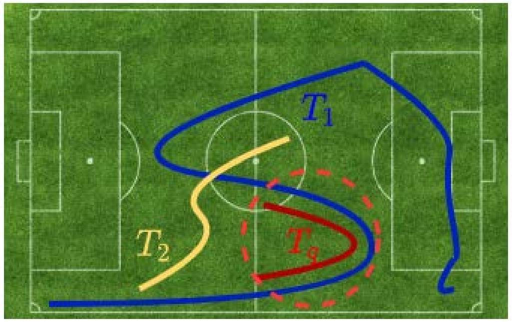
17.14%

82.86%

Outline

- Trajectory Level geospatial data mining
  - Trajectory representation and similarity (ICDE’18, ICDE’19, KDD’19)
    - Trajectory
    - A group of trajectories
  - Sub-trajectory similarity (VLDB’20)
  - trajectory data and its applications in intelligent transport
    - Travel speed estimation (WWW’19)
    - Travel route estimation (ICDE’20)
  - Crowdness estimation

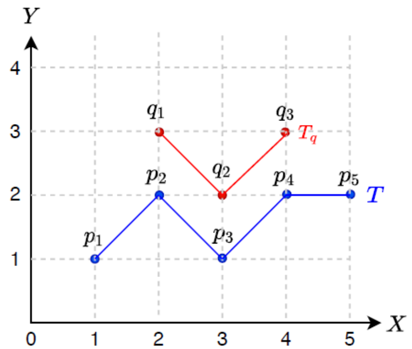
Similar Trajectory Search



T_1 and T_2 are **dissimilar** to T_q

T_1 has a **portion** that is **similar** to T_q

SimSub Problem



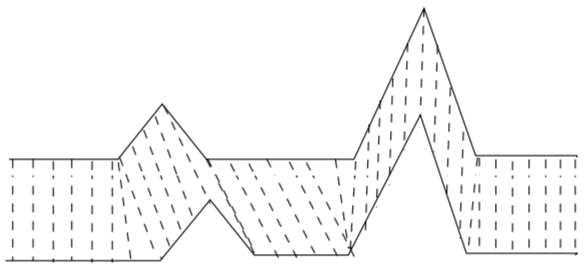
SimSub problem

SimSub problem:
return a portion of a data trajectory (i.e., a subtrajectory), which is the most similar to a query trajectory

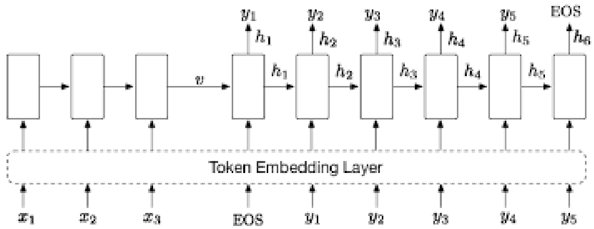
Tq is a query trajectory and T is a data trajectory:

- Return T[2:4] = <p2, p3, p4>
- General framework using any similarity measurement

Trajectory Similarity Measurement

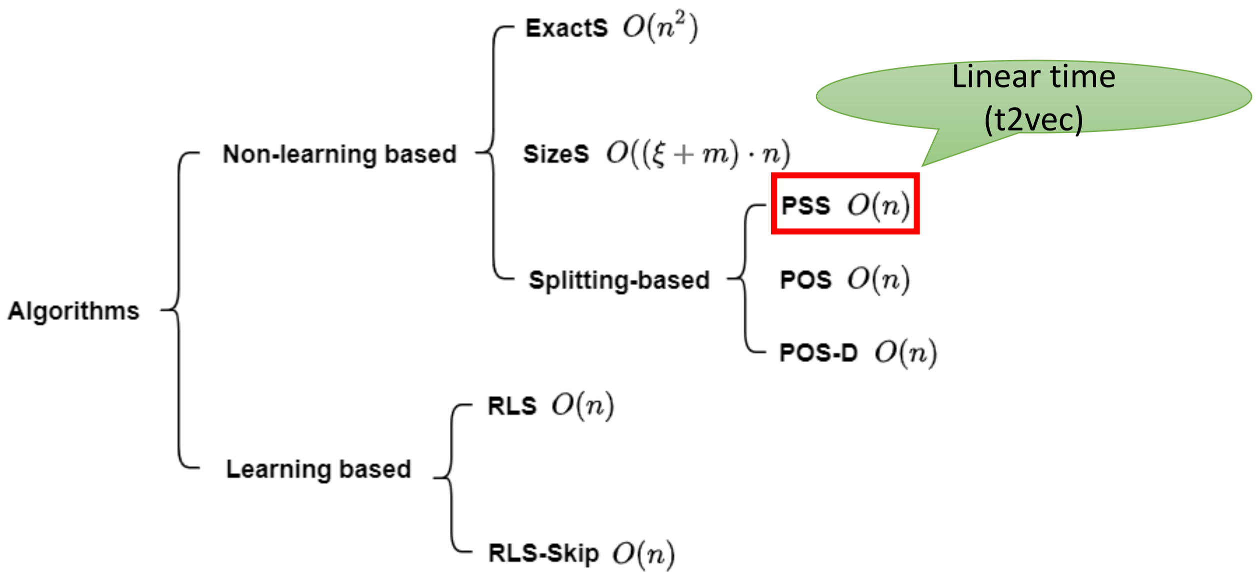


DTW and Frechet:
- **Pairwise matching**
- **Quadratic** time complexity

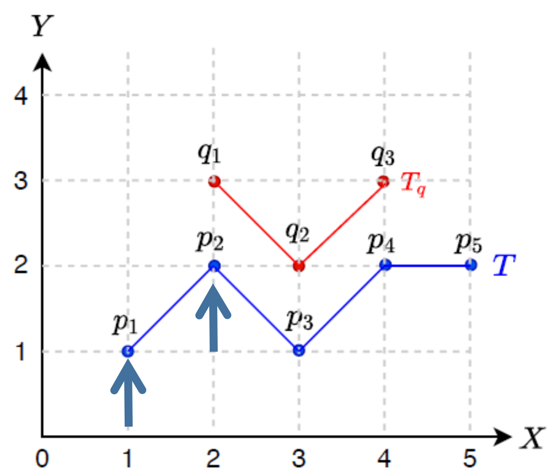


t2vec:
- **Data-driven** (representation learning)
- **Linear** time complexity

Algorithms


Algorithms
Non-learning based
ExactS $O(n^2)$
SizeS $O((\xi + m) \cdot n)$
Splitting-based
PSS $O(n)$
POS $O(n)$
POS-D $O(n)$
Linear time
(t2vec)
Learning based
RLS $O(n)$
RLS-Skip $O(n)$

Prefix-Suffix Search (PSS)



SimSub problem

max(sim_prefix, sim_suffix) > sim_best

Split at current point P_i, it updates the splitting index to i+1

Point	Prefix	Suffix	Split	Splitting index	BEST
P1	T[1,1]	T[1,5]	Yes	2	T[1,5]
P2	T[2,2]	T[2,5]	Yes	3	T[2,2]
P3	T[3,3]	T[3,5]	No	3	T[2,2]
P4	T[3,4]	T[4,5]	No	3	T[2,2]
P5	T[3,5]	T[5,5]	No	3	T[2,2]
Output: BEST=T[2,2]					

max(sim_prefix, sim_suffix) <= sim_best

Algorithms

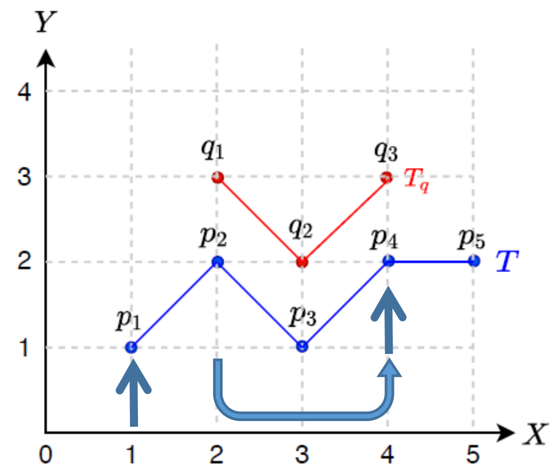
- Algorithms
  - Non-learning based
    - ExactS $O(n^2)$
    - SizeS $O((\xi + m) \cdot n)$
    - Splitting-based
      - PSS $O(n)$
      - POS $O(n)$
      - POS-D $O(n)$
  - Learning based
    - RLS $O(n)$
    - $\boxed{\text{RLS-Skip } O(n)}$

Linear time
(t2vec)

Reinforcement Learning based Search with Skipping (RLS-Skip)

Splitting as an MDP:

- **State: S = (sim_best, sim_prefix, sim_suffix)**
- **Action: Split, No-split, Skip**
- **Transition: from S to S'**
- **Reward: S'.sim_best - S.sim_best**



SimSub problem

Split at current point P_i, it updates the splitting index to i+1

Skip next point

Point	State (best, prefix, suffix)	Action	Splitting index	BEST
P1	(Ø,T[1,1],T[1,5])	Split	2	T[1,5]
P2	(T[1,5],T[2,2],T[2,5])	Skip	2	T[2,2]
P3	-	-	-	-
P4	(T[2,2],T[2,4],T[4,5])	Split	5	T[2,4]
P5	(T[2,2],T[5,5],T[5,5])	No-split	5	T[2,4]
Output: BEST=T[2,4]				

Deep-Q-Network

Experimental Setup

Datasets
Porto (1.7M)
Harbin (1.2M)
Sports (0.2M)

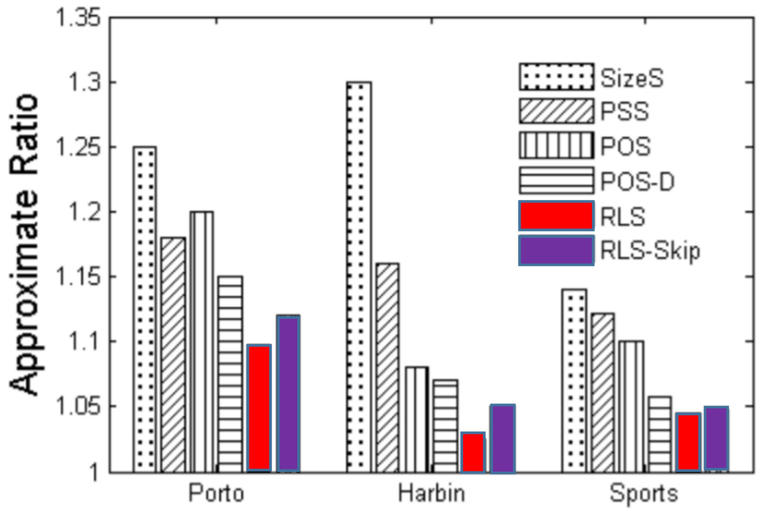
Compared Methods
Proposed algorithms
UCR [1]
Spring [2]
Random-S

Experiments	Performance metrics
Effectiveness	Approximate ratio
	Mean rank
	Relative rank
Efficiency	Running time

[1] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. SIGKDD 2012.
[2] Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. ICDE 2007.

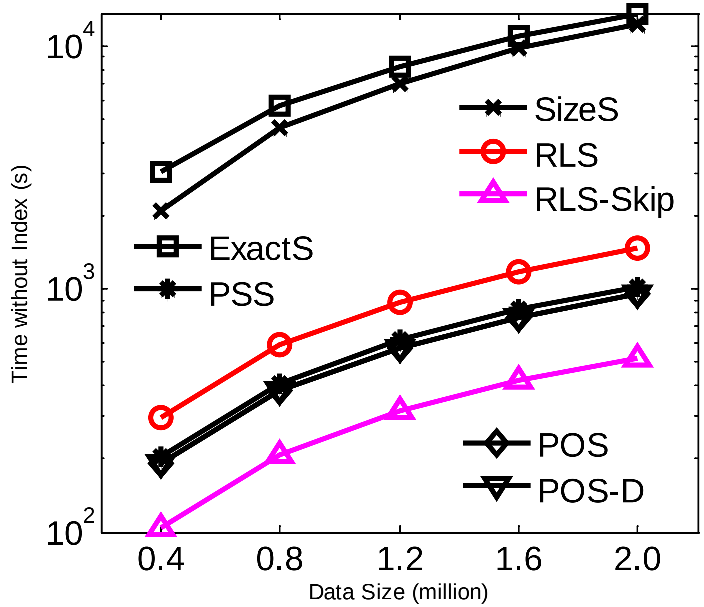
Effectiveness Results

- RLS achieves the best effectiveness



Efficiency Results

- RLS-Skip achieves the best efficiency

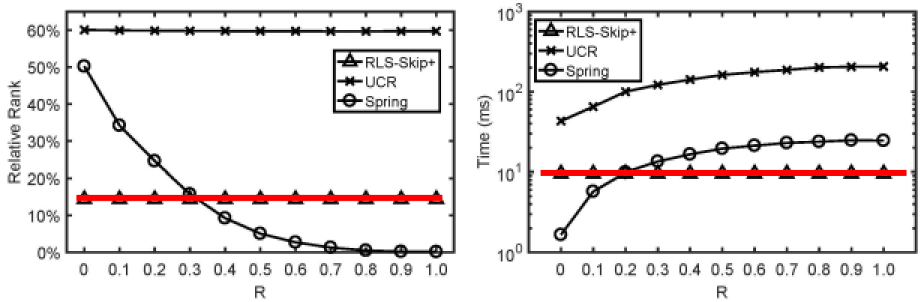


Parameter Study – Skipping Steps k for RLS-Skip

Metrics	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
AR	1.028	1.039	1.042	1.044	1.055	1.069
MR	41.138	56.633	58.077	64.741	70.281	94.356
RR	3.5%	5.4%	5.6%	5.8%	6.3%	8.9%
Time (ms)	55.2	39.8	38.5	35.8	31.8	22.9
Skip Pts	0%	3.1%	13.1%	17.7%	29.5%	47.6%

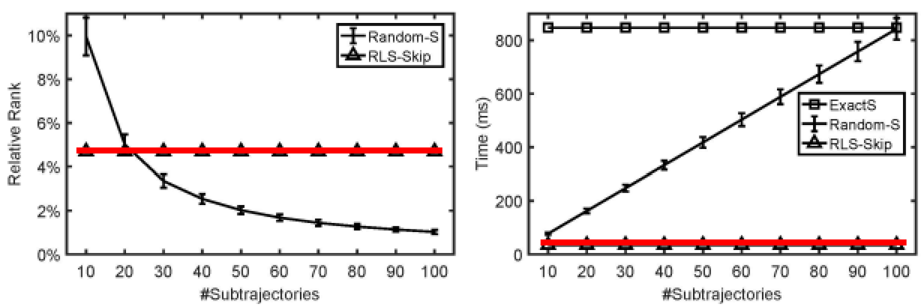
Comparison with UCR, Spring and Random-S

- UCR and Spring



(a) Relative Rank (DTW) (b) Time Cost (DTW)

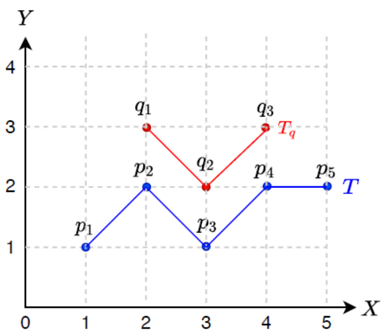
- Random-S



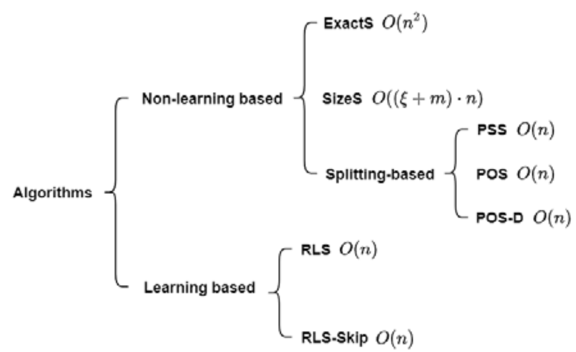
(a) Relative Rank (DTW) (b) Time Cost (DTW)

Conclusion

Problem + **Algorithms** + **Experiments**



SimSub problem



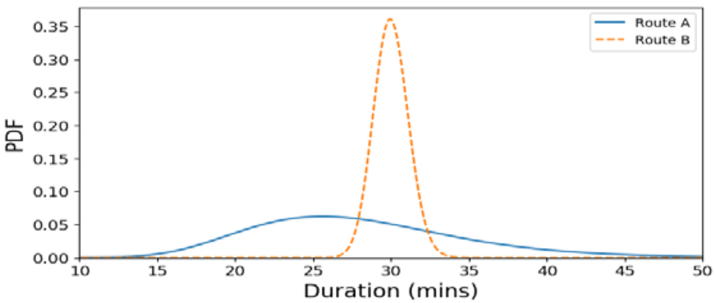
Experiments	Performance metrics
Effectivenes s	Approximate ratio
	Mean rank
	Relative rank
Efficiency	Running time

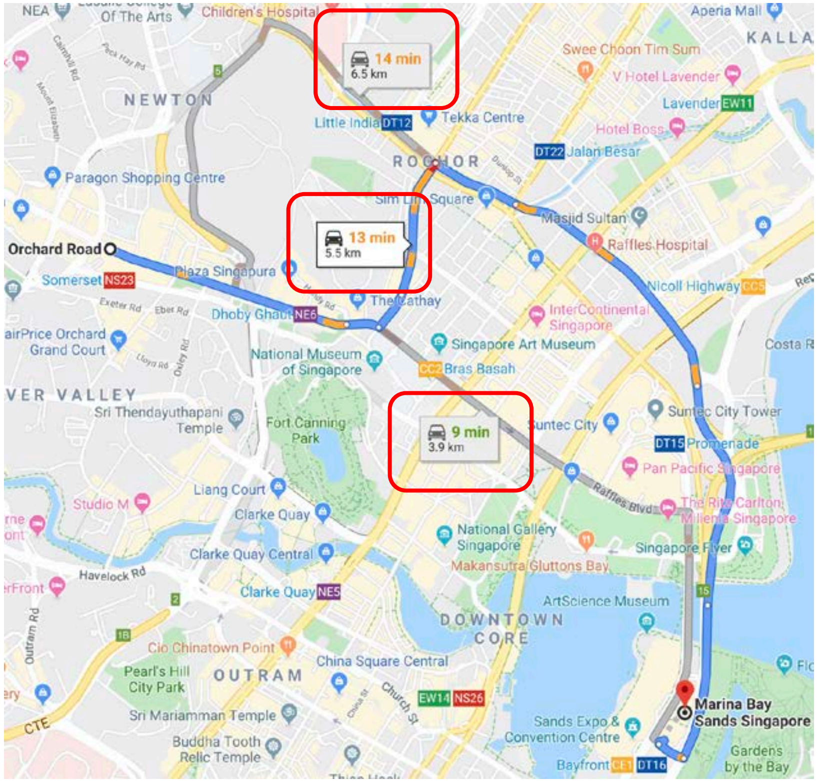
Outline

- Trajectory Level geospatial data mining
  - Trajectory representation and similarity (ICDE’18, ICDE’19, KDD’19)
  - Sub-trajectory similarity (VLDB’20)
  - trajectory data and its applications in intelligent transport
    - Travel speed estimation (WWW’19)
    - Travel route estimation (ICDE’20)
  - Crowdness estimation

Travel time distribution estimation

- Travel time distribution estimation



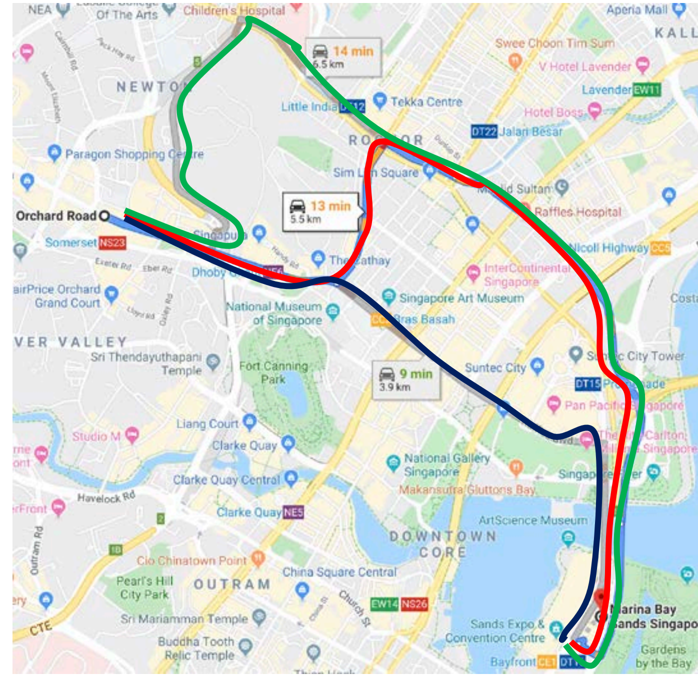


- Given a route on the road network, we aim to learn its travel time distribution (Probability Density Function) with the consideration of real- time traffic.
  - Developed the first deep generative model for the travel time distribution prediction

Travel route inference

- Travel route inference
  - Red? Green? Blue? Others?

- Given the origin and destination, we aim to predict the most likely traveling route on the road network and score the probability of a route being travelled.
  - consider the past travelled route, real-time traffic and destination



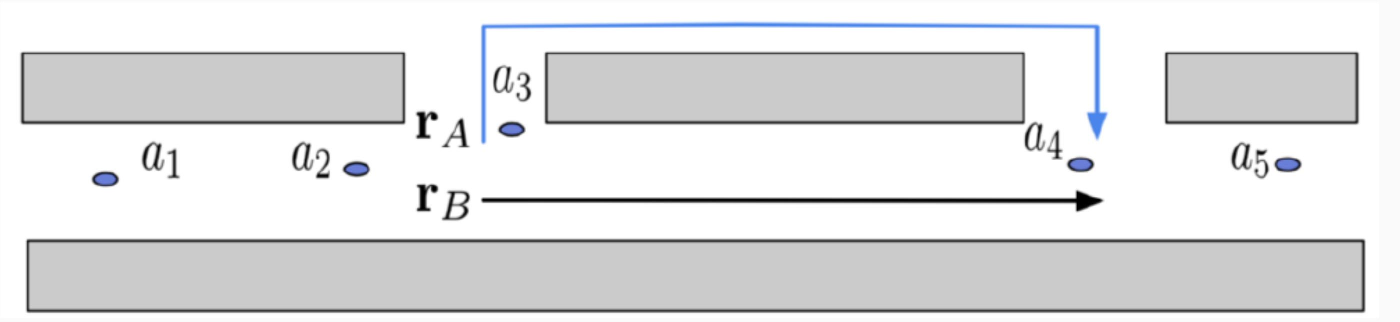
- We develop a novel deep probabilistic generative model – DeepST – to learn the spatial transition patterns, which considers all three key factors.
- We propose a novel adjoint generative model to learn the k-destination proxies.
  - it enables effectively sharing statistical strength across trips
  - the resulting model is robust to inaccurate destinations
- We develop an efficient inference method that scales the model to large-scale datasets within the VAEs framework.

Problem

- In this study, we explore the problem of
  - given the origin and destination,
  - predicting the most likely route on the road netwrok;
  - outputing a probability value to indicate the likelihood of a route being traveled.

- Motivating Example: In taxi dispatch system, the origin and destination are usually given before the start of a trip, and thus predicting the most likely route could help us better arrange the taxi sharing by picking up the potential passengers that are waiting on or nearby the most likely traveled route.

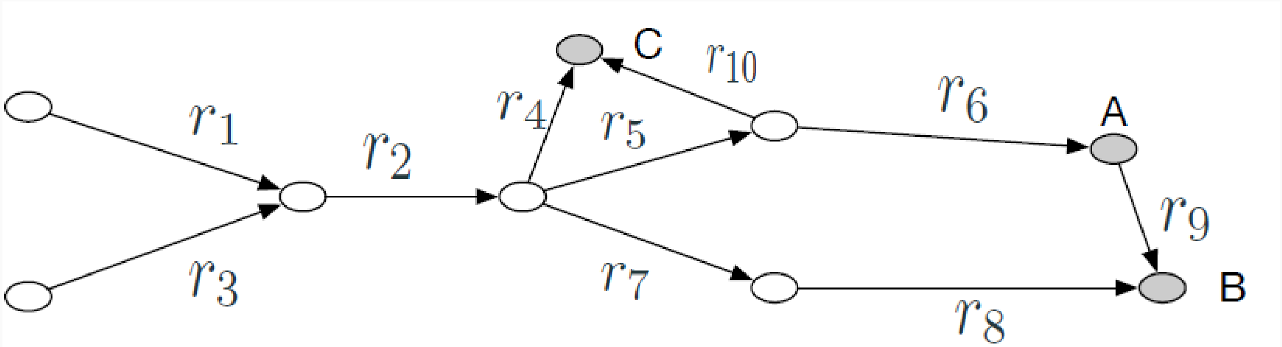
Motivating example



Let's revisit the route recovery problem.

- The problem of inferring the most likely route: $\text{argmax}_{\mathbf{r}} \mathbb{P}(\mathbf{r}|t), \mathbf{r} \in \{\mathbf{r}_A, \mathbf{r}_B\}$.
- Using Bayes rule, $\mathbb{P}(\mathbf{r}|t) \propto p(t|\mathbf{r})\mathbb{P}(\mathbf{r})$.
- We also require to score the spatial transition likelihood $\mathbb{P}(\mathbf{r})$ based on origin and destination.

Key factors in spatial transition modelling



Route	Destination	Frequency
$r_3 \to r_2 \to r_4$	C	400
$r_3 \to r_2 \to r_5 \to r_{10}$	C	100
$r_1 \to r_2 \to r_5 \to r_6$	A	100
$r_1 \to r_2 \to r_7 \to r_8$	B	100
$r_1 \to r_2 \to r_5 \to r_6 \to r_9$	B	100

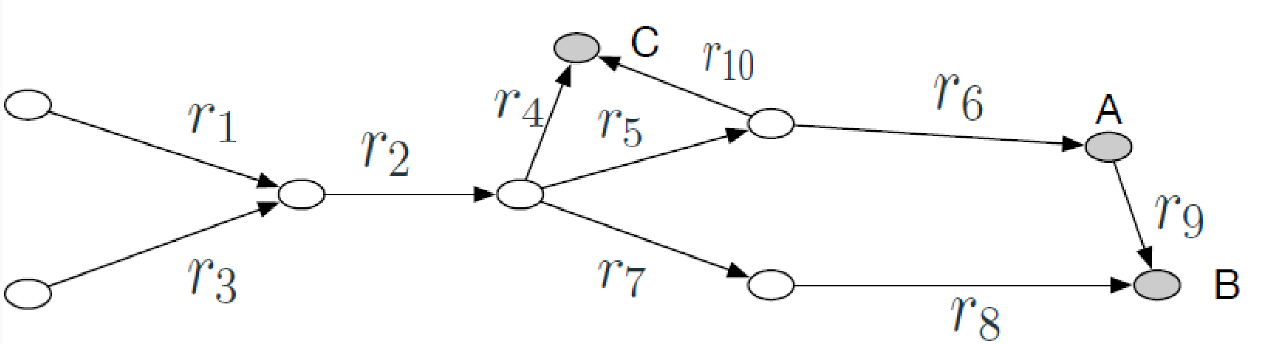
Key factors in spatial transition modelling

The spatial transition patterns demonstrate strong sequential property.

Consider we try to predict a car driving on r_2.

- $\mathbb{P}(r_4|r_2) = 4/8 > \mathbb{P}(r_5|r_2) = 3/8$.
- $\mathbb{P}(r_4|r_1 \to r_2) = 0 < \mathbb{P}(r_5|r_1 \to r_2) = 3/4$.

Key factors in spatial transition modelling



Route	Destination	Frequency
$r_3 \rightarrow r_2 \rightarrow r_4$	C	400
$r_3 \rightarrow r_2 \rightarrow r_5 \rightarrow r_{10}$	C	100
$r_1 \rightarrow r_2 \rightarrow r_5 \rightarrow r_6$	A	100
$r_1 \rightarrow r_2 \rightarrow r_7 \rightarrow r_8$	B	100
$r_1 \rightarrow r_2 \rightarrow r_5 \rightarrow r_6 \rightarrow r_9$	B	100

Key factors in spatial transition modelling

The trip destination has a global impact on the transition.

Consider that a vehicle is driving on r_5 and its destination is C.

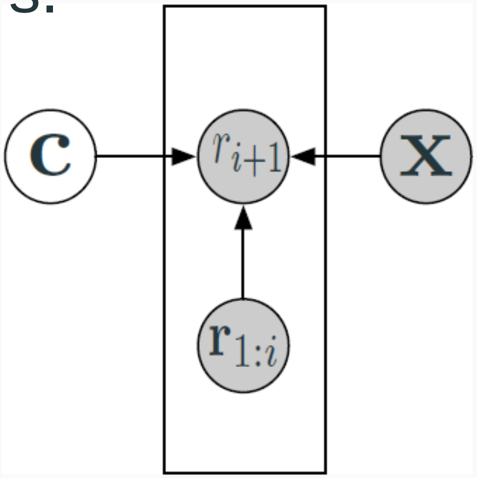
- $\mathbb{P}(r_6|r_5) = 2/3 > \mathbb{P}(r_{10}|r_5) = 1/3$.
- $\mathbb{P}(r_6|r_5, C) = 0 < \mathbb{P}(r_{10}|r_5, C) = 1$.

The route choices are also influenced by the real-time traffic.

The vehicle drivers tend to choose those less congested routes rather than the shortest one.

Generative Process

High level idea: We attempt to generate the observed trips by conditioning on the three key explanatory factors.



- Draw $\mathbf{c} \sim \mathrm{Normal}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$.
- For $i+1$-th road segment, $i \geq 1$
  - Draw $r_{i+1} \sim \mathbb{P}(r_{i+1}|\mathbf{r}_{1:i}, \mathbf{x}, \mathbf{c})$.
  - Draw ending indicator $s \sim \mathrm{Bernoulli}(f_s(r_{i+1}, \mathbf{x}))$.
  - If $s = 0$ then continue else end the generation.

where $f_s(r_{i+1}, \mathbf{x}) = \frac{1}{1+d(r_{i+1}, \mathbf{x})}$.

$$\mathbb{P}(r_{i+1}|\mathbf{r}_{1:i}, \mathbf{x}, \mathbf{c}) = \mathrm{softmax}\left(\alpha^\top f_{\mathbf{r}}(\mathbf{r}_{1:i}) + \beta^\top f_{\mathbf{x}}(\mathbf{x}) + \gamma^\top \mathbf{c}\right)$$

- $f_{\mathbf{r}}(\mathbf{r}_{1:i}) \in \mathbb{R}^{n_{\mathbf{r}}}$ representation of past traveled route $\mathbf{r}_{1:i}$
- $f_{\mathbf{x}}(\mathbf{x}) \in \mathbb{R}^{n_{\mathbf{x}}}$ representation of destination $\mathbf{x}$
- $\alpha \in \mathbb{R}^{n_{\mathbf{r}} \times \mathcal{N}(r_i)}, \beta \in \mathbb{R}^{n_{\mathbf{x}} \times \mathcal{N}(r_i)}, \gamma \in \mathbb{R}^{|\mathbf{c}| \times \mathcal{N}(r_i)}$ are projection matrices

Experimental setup

- **Chengdu dataset**: 33,000 taxis, 15 days, 3 million trips.
- **Harbin dataset**: 13,000 taxis, 28 days, 2.9 million trips.

Dataset statistics.

City	**Chengdu**			**Harbin**		
Measures	min	max	mean	min	max	mean
Distance (km)	1.0	40	4.9	1.1	60	11.4
#road segments	5	85	14	5	102	24

Baseline methods

- DeepST-C is a simplified version of DeepST without considering the impact of real-time traffic.
- RNN is the vanilla RNN that only takes the initial road segment as input.
- CSSRNN [**IJCAI17**] is the state-of-art route decision model.
- MMI is the first-order Markov model.
- WSP always returns the shortest path from the origin road segment to the destination road segment on the weighted road network.
- STRS [**KDD18**] is the state-of-art route recovery method comprising of a travel time inference module and a spatial transition inference module.

Overall performance

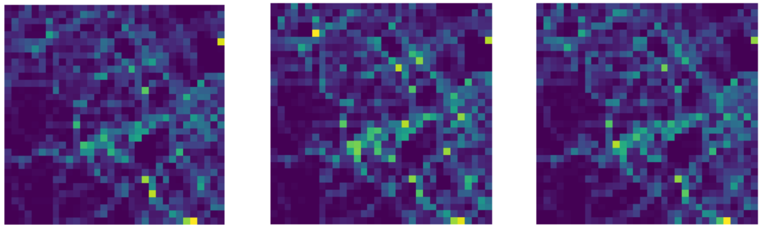
Overall performance.

City	Chengdu					
Method	DeepST	DeepST-C	CSSRNN	RNN	MMI	WSP
recall@n	**0.637**	0.626	0.577	0.409	0.318	0.431
accuracy	**0.612**	0.601	0.556	0.389	0.281	0.431

City	Harbin					
Method	DeepST	DeepST-C	CSSRNN	RNN	MMI	WSP
recall@n	**0.397**	0.385	0.336	0.261	0.202	0.267
accuracy	**0.374**	0.366	0.313	0.172	0.154	0.267

Crowd density prediction

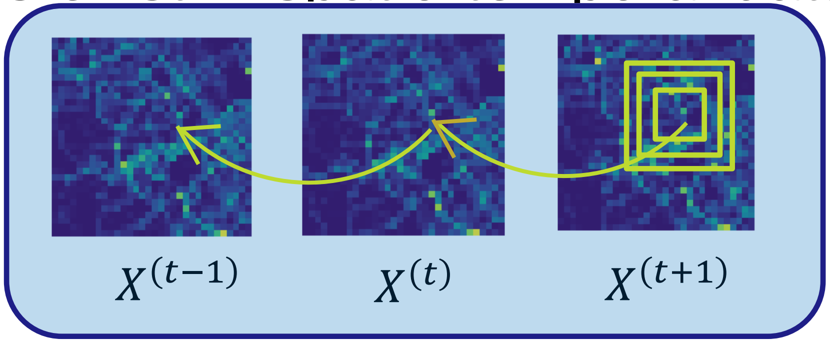
- Crowd density prediction is an important problem in geo-spatial domain
  - Traffic management
  - Urban planning
  - Telecommunication networks
- Large scale real-time geo-tagged data is available today:
  - Vehicles equipped with GPS sensors
  - People carrying mobile phones, etc



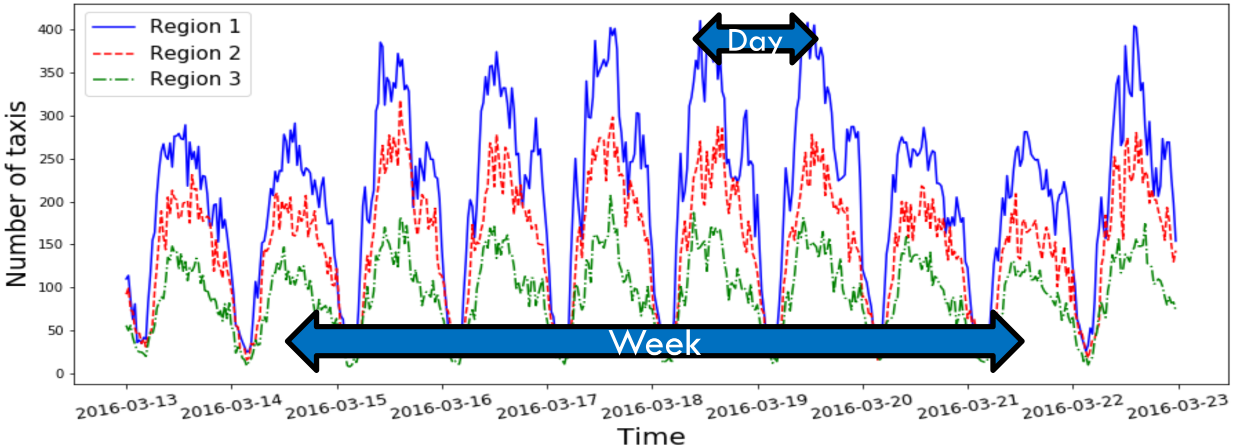
$X^{(t-1)}$ $X^{(t)}$ $X^{(t+1)}$

Spatial-Temporal Prediction

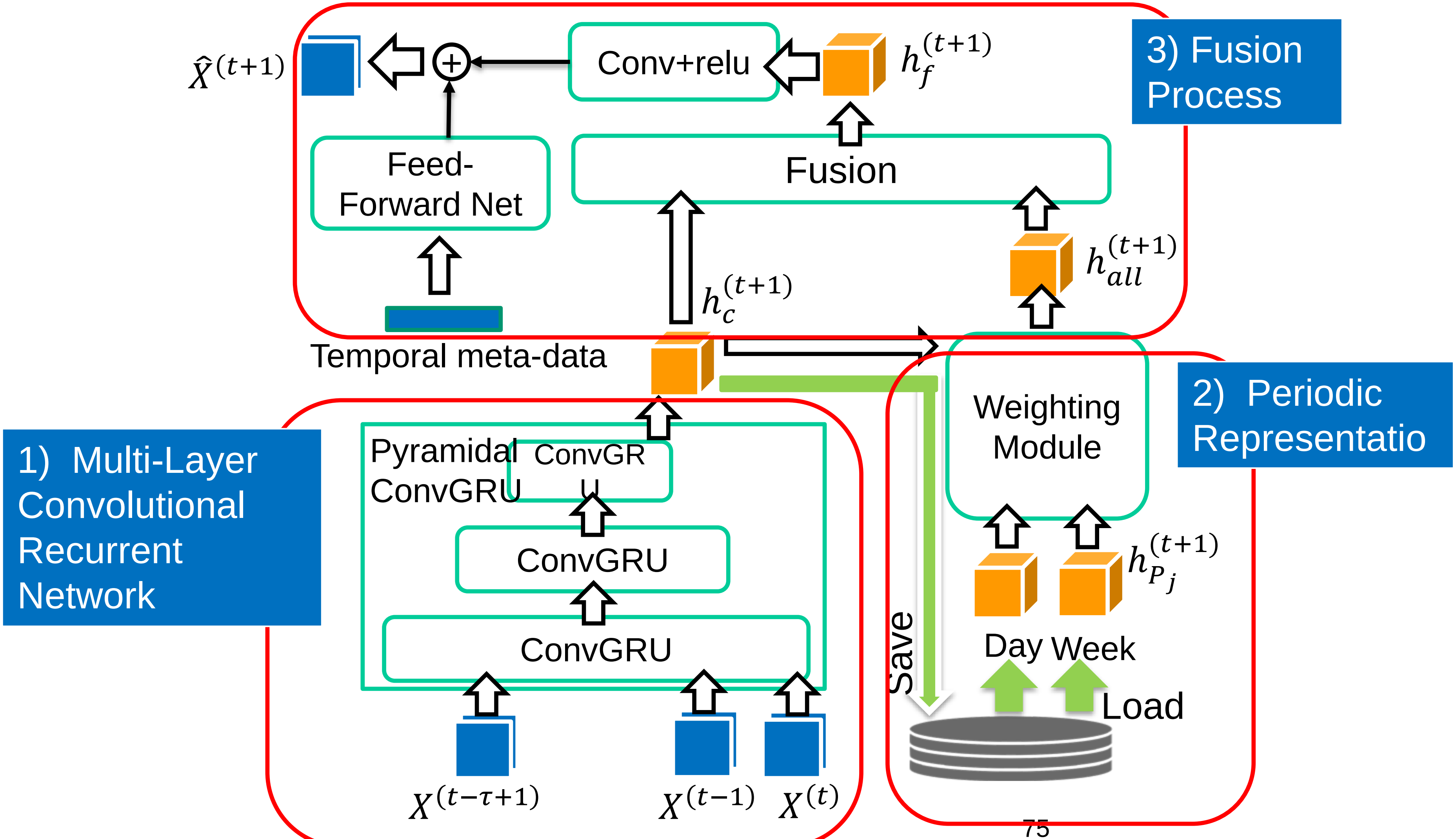
- Two types of dependencies exist in spatio-temporal data
  - Temporal dependencies
  - Spatial dependencies



- Periodic patterns over different time scale (days, weeks) exist in many spatio-temporal time-series data



Proposed Model



Smart Nation Applications

GeoSpatial Data Mining

- POI recommendation & prediction
- Interactive exploration geospatial data
- Knowledge graph for locations
- Trajectory representation and similarity
- Speed, travel time, route prediction
- Region search, (e.g., burst region)
- Region exploration (topic, crowdness)

Querying and indexing spatio-temporal data

- Snapshot queries (OLTP, OLAP)
- Continuous queries

Distributed streaming systems

- Distributed load balance
  Distributed materialized view
- Index &query optimizer
- Machine learning techniques

Big static/streaming geo-spatial + X (e.g., text, temporal) data

foursquare yelp Grab Google places twitter Hi, stranger Instagram Fast beautiful photo sharing

Acknowledgement to my students and collaborators: Yile Chen, Cheng Long, Xiucheng Li, Yiding Liu, Zheng Wang, Di Yao, Kaiqi Zhao.

Thank You !

Q & A?

Demo URL: http://spatialkeyword.sce.ntu.edu.sg/index.html#