Effective and Efficient Sports Play Retrieval with Deep Representation Learning

Zheng Wang
Nanyang Technological University
Singapore
wang_zheng@ntu.edu.sg

Cheng Long
Nanyang Technological University
Singapore
c.long@ntu.edu.sg

Gao Cong
Nanyang Technological University
Singapore
gaocong@ntu.edu.sg

Ce Ju
Intelligent Driving Group, Baidu Inc.
Beijing, China
juce@baidu.com

ABSTRACT

With the proliferation of commercial tracking systems, sports data is being generated at an unprecedented speed and the interest in sports play retrieval has grown dramatically as well. However, it is challenging to design an effective, efficient and robust similarity measure for sports play retrieval. To this end, we propose a deep learning approach to learn the representations of sports plays, called play2vec, which is robust against noise and takes only linear time to compute the similarity between two sports plays. We conduct experiments on real-world soccer match data, and the results show that our solution performs more effectively and efficiently compared with the state-of-the-art methods.

CCS CONCEPTS

• **Information systems** → *Data mining*; • **Computing methodologies** → *Knowledge representation and reasoning*.

KEYWORDS

Sports play retrieval; deep representation learning; paly2vec

ACM Reference Format:
Zheng Wang, Cheng Long, Gao Cong, and Ce Ju. 2019. Effective and Efficient Sports Play Retrieval with Deep Representation Learning. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330927

1 INTRODUCTION

Nowadays, it becomes a common practice to track the moving agents in a sports game (e.g., the players and the ball in a soccer game) using cameras and/or GPS devices. For example, the SportVU system by STATS LLC, which is an optical tracking system based on cameras, has been used by professional sports leagues such as France's Ligue de Football Professionnel (LFP) and American National Basketball Association (NBA). The data collected by these tracking systems can be represented as the trajectories of the players and the ball in a game, i.e., it embeds both spatial and temporal features of a game. Hence, it is usually termed as spatiotemporal sports data and has been used in many sports analytics tasks such as team formation detection [4, 16], movement pattern mining [8, 18], tactics discovery [8], score prediction [2, 30], and similar play retrieval [24].

Similar play retrieval is a process of finding those plays from a database that are similar to a query play, where a *play* corresponds to a fragment of a game and has its duration varying from seconds to minutes. It is widely used in some emerging sports applications such as ESPN and Team Stream to recommend similar plays to sports fans. Besides, it could help sports club managers and coaches to improve team tactics when preparing for an upcoming match [8].

The core problem of similar play retrieval is measuring the similarity between two plays, which is non-trivial because each play involves *multiple* trajectories. Existing solutions for this problem all adopt a two-step approach: (1) it *aligns* the trajectories in one play to those in the other; and (2) it computes the similarity between each pair of trajectories that have been aligned to each other using some trajectory similarity metrics such as the Dynamic Time Warping (DTW) [29] and then sums up all similarities to be one between the two plays [24]. Two strategies have been considered for the alignment step [24]. The first one is an enumeration-based strategy which (conceptually) considers all possible alignments and picks the one with the best similarity score. In practice, the method with this alignment strategy could be implemented by finding the optimal matching between the two sets of trajectories using algorithms such as the *Hungarian* algorithm, where the weight between a pair of trajectories is set to be the similarity between them. The second one is a role-based strategy which first detects the role of each player, which is hidden, and then aligns the trajectories of two players who share their roles. Here, the role of a player detected may have semantics such as center forward, midfield, etc. in a soccer game and could change from time to time throughout the game.

While these existing solutions have some merits in transforming the original problem for plays to one for trajectories with the alignment strategy, they are insufficient in three aspects.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330927

The first one is the effectiveness. These methods rely on the assumption that human beings would check individual pairs of trajectories separately and then combine their perceptions of similarity on trajectories together using a sum function, which, however, remains not justified. For example, these methods assume that each pair of trajectories contributes equally to the final similarity between two plays, but it is more intuitive to assign more weights to those pairs of trajectories that are closer to the ball.

The second one is the efficiency. These methods all involve the procedure of computing the similarity between two trajectories, which has a time cost at least $O(n^2)$ for those well-known trajectory metrics (including the DTW one) where n is the length of the longer trajectory. Since a typical similar play retrieval scenario would usually need to compute the similarity between the query play and many plays in the database, which is huge in practice (e.g., a typical soccer game involves about 4.14 million sampled positions and one season of soccer games in the English Premier League involves about 1.57 billion sampled positions), this quadratic time complexity would impose a big challenge on the efficiency.

The third one is the robustness (against sampling errors and measurement errors). Recall that the spatiotemporal sports data is collected by sampling the locations of the players and the ball with devices such as GPS. Two types of errors are inevitable in the data, namely the errors due to the sampling nature and those due to the measurement of devices. Existing methods use the locations in the form of coordinates and thus some minor changes on the coordinates may result in an obvious change on the similarity, i.e., they are not robust against the errors.

In this paper, we propose to learn representations of plays in a low-dimensional space using deep models, which we call *play2vec*, such that the (Euclidean) distances in the space capture the similarities among the plays well. The core idea of our approach is to treat a play as a sequence of play segments with uniform durations and then design a denoising sequential encoder-decoder (DSED) model for extracting a feature vector from the sequence. There is a gap that needs to close up to make this idea work since a play segment, which corresponds to a fragment of a play, has the same form as a play - it consists of a set of multiple trajectories (of shorter lengths) while an encoder-decoder model typically accepts inputs in the form of vectors. We achieve this by embedding each play segment as a vector. Specifically, we treat each play segment as a word or token and each play as a sentence in a natural language context and extend the Skip-Gram with Negative Sampling (SGNS) model [17] to embed the words (or play segments) as vectors. Again, there is a gap here since a play segment is in a continuous space (there exists an infinite number of possible trajectories) while a word is in a discrete one. We close this gap by (1) mapping each play segment to a binary matrix, where an entry of the matrix is set to be 1 if its corresponding cell in a grid that partitions the pitch is traveled through by some trajectories in the play segment and 0 otherwise; and (2) assigning one token to those matrices that are similar (for restricting the token space size). In summary, our approach first maps all play segments to tokens with a spatial grid, then embeds the tokens as vectors by extending the SGNS model, and then extracts a feature vector from each play with the DSED model that is designed to fit our context.

Our new method has obvious advantages over existing ones in the aforementioned three aspects. Regarding the effectiveness, our method is based on the popular encoder-decoder deep model which is widely known to perform well in extracting features from sequential data. Regarding the efficiency of computing the similarity of two plays, our method runs in $O(n + d)$ time while existing ones have time complexities at least $O(n^2)$, where n is the length of the longest trajectory in a play and d is the size of a learned feature vector which is small. Regarding the robustness, our method involves two mechanisms to deal with sampling errors and measurement errors. First, in the step of mapping play segments to tokens, a grid is used such that it is not sensitive to errors. Second, in the encoder-decoder model, the data is first injected with some noises and then denoised for training which would help mitigate the problems caused by errors.

In summary, the main contribution includes: (1) we develop an unsupervised deep learning model, play2vec, to learn representations of plays, which is superior over existing ones in aspects of effectiveness, efficiency, and robustness; and (2) we perform extensive experiments on real-world soccer data, which show that our method consistently outperforms the state-of-the-art in terms of effectiveness and runs faster than existing methods by over one order of magnitude. We further conduct a user study which validates our approach with expert knowledge.

Organization. We review the related work in Section 2 and give the problem definition in Section 3. We present our play2vec method in Section 4, report our experimental results in Section 5 and conclude the paper in Section 6.

2 RELATED WORK

2.1 Sports Data Analytics

The conventional methods for sports play retrieval are based on "keywords", which however requires the data to be annotated with keywords and users to have necessary prior knowledge on keywords. Most germane to our work is the work by Sha et al. [24, 25], which measures the similarity between two plays by first aligning trajectories from two plays (based on the extracted roles of trajectories) and then aggregating the similarities between aligned trajectories as one between the two plays. Probst et al. [20] focus on queries such as region queries based on spatiotemporal sports data. Di et al. [9] propose to extract features from sports plays by using CNNs on the visual representations of trajectories and use the extracted features together with some other features to learn a rankSVM model for serving users with specific preferences (conveyed with click-through data). Other types of sports analytics include those of detecting team formations [4, 16], identifying spatial patterns of movement [8, 18], analyzing sports videos [15] and sports prediction [2, 30]. The work [10] gives a more detailed survey on spatiotemporal sports data analytics.

2.2 Measuring Trajectory Similarity

The problem of measuring the similarity between trajectories (time series in general) has been studied extensively. DTW [29] is the first attempt towards solving the local time shift issue for computing trajectory similarity. Frechet distance [1] is a classical similarity measure that treats each trajectory as a spatial curve and takes

Table 1: Notations and meanings.

Notation	Meaning
$\mathcal{D}$	database of plays
$\mathcal{P}$	play
$\mathcal{M}$	segment matrix
$\mathcal{V}$	sports corpus
$\mathcal{T}$ ($\tilde{\mathcal{T}}$)	segment token (corrupted version)
$\boldsymbol{v}_{\mathcal{T}}$	distributed representation of $\mathcal{T}$
$\boldsymbol{v}$	embedding vector

into account the location and order of the sampling points. ERP [5] and EDR [6] are proposed to further improve the ability to capture spatial semantics in trajectories. Nevertheless, these methods are mainly based on alignment of matching sample points, and thus they are inherently sensitive to noise and varying sampling rates which exist commonly in trajectory data. To address this issue, Su et al. [26] propose an anchor-based calibration system that aligns trajectories to a set of fixed locations. Ranu et al. [21] formulate a robust distance function called EDwP to match trajectories under inconsistent and variable sampling rates. These similarity measures are usually based on the dynamic programming technique to identify the optimal alignment which leads to $O(n^2)$ computation complexity, where n is the length of the trajectories. More recently, Li et al. [14] propose to learn representations of trajectories in the form of vectors and then measure the similarity between two trajectories as the Euclidean distance between their corresponding vectors and Yao et al. [28] employ deep metric learning to accelerate trajectory similarity computation. The main difference between our study and these studies is that our problem is on plays (which correspond to sets of multiple trajectories) while these existing ones are on single trajectory. Another related study is one studying trajectory set similarity on road networks by He et al. [11], in which the idea of the Earth Mover's Distance (EMD) is leveraged to capture both spatial and temporal characteristics of trajectories.

2.3 Representation Learning

Inspired by the success of word2vec, the idea of representation learning [3] is widely used for many tasks such as natural language processing [13] and graph embedding [19]. The Skip-Gram with Negative Sampling (SGNS) model [17] is one of common methods of word2vec which is based on the assumption in linguistics that words frequently occurring in a sentence tend to share more statistical information. Seq2Seq based learning model has achieved good performance on spatiotemporal data [14, 27]. The ability to capture the local spatial correlation makes it inherently applicable to various downstream analysis tasks. Our proposed model is inspired by the Seq2Seq model and the SGNS architecture. In this paper, we propose to learn representations of sports plays, which are quite different from those targeted in previous studies. Moreover, to accelerate the training of our model play2vec, we design a method to generate training data with hard negative sampling. We also use a grid structure that is robust to noise and varying sampling rates.

3 PROBLEM DEFINITION

We model a sports *game* by the movements of the objects involved in the game (e.g., in a soccer game, the objects include 22 players from two teams and also a ball). The movement of an object is usually captured by sampling its locations at a certain frequency with tracking technologies such as those based on GPS devices. As a result, the movement of an object corresponds to a sequence of time-stamped locations, which is called a *trajectory*. A trajectory has its form of (x_1, y_1, t_1), (x_2, y_2, t_2), ..., where (x_i, y_i) is the i^{th} location and t_i is the time stamp of the i^{th} location. Therefore, a play corresponds to a set of multiple trajectories.

A *play* corresponds to a fragment of a game and has its duration varying from seconds to minutes, depending on users' needs. The concept of play provides the flexibility of searching finer-grained games (i.e., game fragments). Same as a game, a play corresponds to a set of multiple trajectories (with shorter lengths).

Given a database of plays $\mathcal{D}$, we aim to learn a vector representation $\boldsymbol{v} \in \mathbb{R}^d$ for each play $\mathcal{P} \in \mathcal{D}$ in a d-dimensional space such that the similarities among plays are well captured by the Euclidean distances in the d-dimensional vector space, i.e., for any two plays, if they are similar, the distance between their vectors would be small.

The notations that are frequently used throughout the paper are given in Table 1.

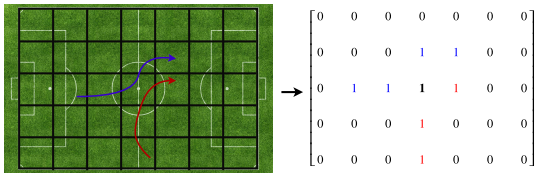
4 METHODOLOGY

In this part, we introduce our deep learning method, play2vec, for learning vector presentations of plays. The core idea is that we break each play into a sequence of non-overlapping segments of a fixed duration, each called a *play segment*, and then design an encoder-decoder model to extract the features from the sequence as a vector. Specifically, our method first builds a corpus $\mathcal{V}$ based on all play segments (Section 4.1), then adopts the Skip-Gram with Negative Sampling (SGNS) model [17] for learning distributed representations of the tokens in $\mathcal{V}$ (Section 4.2), and eventually glues all distributed representations of the play segments in a play, yielding a vector representation $\boldsymbol{v}$ using the Denoising Sequential Encoder-Decoder (DSED) model that is designed in this paper (Section 4.3).

4.1 Building a Sports Corpus

First, we introduce a process of mapping a play segment to a binary matrix with the help of a grid. Specifically, we divide the pitch into a grid with equal cell size γ, for which we would have a corresponding matrix called *segment matrix*, i.e., each cell in the grid has a corresponding entry in the matrix. Given a play segment, which consists of a set of trajectories, we set to 1 all those entries whose corresponding cells are traveled through by the trajectories and 0 the remaining entries. An example is shown in Figure 1 for illustration. Note that in this example, the 5×7 grid map is determined by the cell size, which is set empirically in Section 5.2.4.

Since each segment matrix has binary values only, the number of possible segment matrices is limited. A simple strategy is to create one unique token for each possible segment matrix. Nevertheless, with this strategy, the resulting corpus could be big, which may affect the effectiveness of the representation learning afterwards. Thus, we propose to scan the segment matrices one by one and for each segment matrix, we create a new token only if it is dissimilar from those segment matrices that have been scanned to a certain extent, where we use the Jaccard index for measuring the similarity



(a) soccer pitch (b) segment matrix $\mathcal{M}$

Figure 1: Mapping a play segment to a matrix. The soccer pitch is divided into 5×7 grid map. There are two trajectories with the color red and blue in the soccer pitch.

Algorithm 1 Building the Corpus

Input: $\mathcal{D}$: the database of plays; ϕ: Jaccard index threshold
Output: The sports corpus $\mathcal{V}$

1: initialize $\mathcal{V} \leftarrow \emptyset$; $id \leftarrow 1$;
2: **for** each play segment of a play in $\mathcal{D}$ **do**
3: $\mathcal{M} \leftarrow$ the segment matrix mapped from the play segment;
4: **if** $|\mathcal{V}| = 0$ **then**
5: $\mathcal{V} \leftarrow \mathcal{V} \cup (\mathcal{M}, \mathcal{T}_{id})$, $id = id + 1$;
6: **end if**
7: $(\mathcal{M}', \mathcal{T}') \leftarrow \underset{(\mathcal{M}'', \mathcal{T}'') \in \mathcal{V}}{\arg\max} \mathcal{J}(\mathcal{M}, \mathcal{M}'')$;
8: **if** $\mathcal{J}(\mathcal{M}, \mathcal{M}') > \phi$ **then**
9: $\mathcal{V} \leftarrow \mathcal{V} \cup (\mathcal{M}, \mathcal{T}')$;
10: **else**
11: $\mathcal{V} \leftarrow \mathcal{V} \cup (\mathcal{M}, \mathcal{T}_{id})$; $id = id + 1$;
12: **end if**
13: **end for**

between two segment matrices $\mathcal{M}$ and $\mathcal{M}'$, which is defined as follows.

$$\mathcal{J}(\mathcal{M}, \mathcal{M}') := \frac{m_{11}}{m_{01} + m_{10} + m_{11}}, \tag{1}$$

where m_{11} means the total number of attributes where $\mathcal{M}$ and $\mathcal{M}'$ both have a value of 1, m_{01} (or m_{10}) means the total number of attributes where $\mathcal{M}$ is 0 (or 1) and $\mathcal{M}'$ is 1 (or 0).

Algorithm 1 presents the steps of building sports corpus. It first initializes two variables: $\mathcal{V}$ which is the target sports corpus and id which is the index of tokens (line 1). It then has a for loop of scanning the play segments (lines 2-13). Within the loop, it first computes the segment matrix for the play segment currently being processed (line 3). Then, it inserts the first segment matrix and its corresponding token into $\mathcal{V}$ (lines 4-6). It then finds the pair $(\mathcal{M}', \mathcal{T}')$ such that $\mathcal{M}'$ is the most similar to $\mathcal{M}$ among those maintained in $\mathcal{V}$ under the Jaccard index (line 7). If the similarity is above a threshold ϕ, $\mathcal{M}$ is assigned with the same segment token as $\mathcal{M}'$ (lines 8-9); otherwise, $\mathcal{M}$ is assigned with a new segment token (lines 10-11).

4.2 Learning Distributed Representations

In this part, we introduce a method for embedding the play segments (or their corresponding tokens) as d'-dimensional real-valued vectors. Inspired by the success of word2vec techniques in natural language processing, we adopt the Skip-Gram with Negative Sampling (SGNS) model for this task. The effectiveness of a SGNS model depends on how good the context of a token is modeled. The segment tokens occurring in the same context tend to have similar sports scenes. Hence, we use the consecutive segment tokens after and before a token as the forward-looking and backward-looking context of the token, respectively.

In this part, we abuse the notation $\mathcal{V}$ to denote the set containing all segment tokens of the play segments involved in a database of plays. Consider a play $\mathcal{P}$ which involves L play segments. Correspondingly, there is a sequence of L segment tokens which we assume are $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_L$. Then, the m-size window context of a segment token $\mathcal{T}_t$ $(m + 1 \leq t \leq L - m)$, denoted by $c^m(\mathcal{T}_t)$, corresponds to $< \mathcal{T}_{t-m}, \ldots, \mathcal{T}_{t-1}, \mathcal{T}_{t+1}, \ldots, \mathcal{T}_{t+m} >$, where we say $\mathcal{T}_t$ is a target segment token and each token in $c^m(\mathcal{T}_t)$ a context segment token. Note that a segment token could be a target one and also a context one (with others as the target ones), i.e., a segment token has two types of roles, namely a target segment token and a context segment token. Given a target token $\mathcal{T}$ and a context token C, we say the token-context pair $(\mathcal{T}, C)$ is positive if $C \in c^m(\mathcal{T})$ and negative otherwise. Considering that a segment token has two types of role, we define for each segment token a vector $\boldsymbol{v}_{\mathcal{T}} \in \mathbb{R}^{d'}$ for cases it corresponds to a target segment token and a vector $\boldsymbol{v}_C \in \mathbb{R}^{d'}$ for cases it corresponds to a context segment token, where d' is the embedding dimension. We aim to learn these vectors such that we could infer those context segment tokens from a target one with the maximum probability.

We explain the training data and also the loss next. The training data consists of a set of training samples. Each training sample consists of one positive token-context pair $(\mathcal{T}, C)$ (i.e., $C \in c^m(\mathcal{T})$) and k negative pairs $(\mathcal{T}^i, C)$, where $\mathcal{T}^i$ for $i = 1, 2, ..., k$ is drawn from a segment token distribution $Q(\mathcal{T})$. Specifically, $Q(\mathcal{T})$ is a α-smoothed unigram distribution,

$$Q(\mathcal{T}) := \frac{f(\mathcal{T})^\alpha}{\sum_{\mathcal{T}' \in \mathcal{V}} f(\mathcal{T}')^\alpha}, \tag{2}$$

where $\alpha \in [0, 1]$ and $f(\mathcal{T})$ is the frequency of the segment token $\mathcal{T}$ appearing in the corpus.

The loss is explained as follows. For a token-context pair $(\mathcal{T}, C)$, we model the probability that the pair is positive as $\sigma\Big((\boldsymbol{v}_{\mathcal{T}})^T \cdot \boldsymbol{v}_C\Big)$ (i.e., a sigmoid function is used with its input as the dot product between the vectors of the segment tokens) and the pair is negative as $1 - \sigma\Big((\boldsymbol{v}_{\mathcal{T}})^T \cdot \boldsymbol{v}_C\Big)$. We then define the loss over one training sample using negative log probabilities as follows.

$$\mathcal{L}(\mathcal{T}, C, \mathcal{T}^i) := -\log \sigma\Big((\boldsymbol{v}_{\mathcal{T}})^T \cdot \boldsymbol{v}_C\Big) - \sum_{i=1}^{k} \log \sigma\Big(-(\boldsymbol{v}_{\mathcal{T}^i})^T \cdot \boldsymbol{v}_C\Big), \tag{3}$$

where $(\mathcal{T}, C)$ is the positive token-context pair and $(\mathcal{T}^i, C)$, for $1 \leq i \leq k$, are the k negative pairs in the training sample. The overall loss function $\mathcal{L}$ is defined by aggregating the losses on all training samples.

The SGNS of the segmentation tokens yields a distributed representation $\boldsymbol{v}_{\mathcal{T}}$ for each $\mathcal{T} \in \mathcal{V}$. The learning algorithm is shown in Algorithm 2. The training starts from a random vector as the initialized embedding, and it will be continuously updated by the stochastic gradient descent to push the target segment token close to the context in the positive pairs and away from the context in the negative pairs.

Algorithm 2 Learning Distributed Representation

Input: $\mathcal{V}$: sports corpus; d': the embedding dimension; m: context window size; k: the number of negative samples; α: learning rate
Output: The learned distributed representation $\boldsymbol{v}_{\mathcal{T}}$ for $\mathcal{T} \in \mathcal{V}$

while improvement on validation set **do**
  for each $\mathcal{T} \in \mathcal{V}$ **do**
    sample a positive pair $(\mathcal{T}, C)$, where $C \in c^m(\mathcal{T})$ is picked randomly;
    sample k negative pairs $(\mathcal{T}^i, C)$, where $\mathcal{T}^i \sim Q(\mathcal{T}), 1 \leq i \leq k$;
    $\boldsymbol{v}_{\mathcal{T}} \leftarrow \boldsymbol{v}_{\mathcal{T}} - \alpha \nabla_{\boldsymbol{v}_{\mathcal{T}}} \mathcal{L}(\mathcal{T}, C, \mathcal{T}^i)$;
  end for
end while

4.3 Bottom-up Gluing

In this section, we aim to glue the distributed representations of the segment tokens up together to achieve a comprehensive representation of the play. We propose a new algorithm framework called the Denoising Sequential Encoder-Decoder (DSED). The intuition is that we try to maximize the probability of recovering the most likely real (or clean) tokens from the corrupted initial inputs. For a given play segment and its corresponding token $\mathcal{T}$, we generate a corrupted version of the token, denoted by $\tilde{\mathcal{T}}$, as follows. We scan the locations of the trajectories contained in the play segment time stamp by time stamp, and at each time stamp, we keep the locations with a pre-set probability (and drop the locations with one minus the probability and continue to the next time stamp) and in the case we keep the locations, we sample for each location a noise following a normal distribution $\mathcal{N}(0, 1)$ and add the noise into the location. We then get a new set of tokens based on the corrupted trajectories.

The architecture of the model is presented in Figure 2. We define the encoding hidden representation $\boldsymbol{h}_{\boldsymbol{t}}^{\boldsymbol{enc}}$ at each time step t, i.e. $\boldsymbol{h}_{\boldsymbol{t}}^{\boldsymbol{enc}} := LSTM_{\theta}^{enc}(\boldsymbol{v}_{\tilde{\mathcal{T}}_t}, \boldsymbol{h}_{\boldsymbol{t-1}}^{\boldsymbol{enc}})$. The encoding hidden vector at the last time step $\boldsymbol{h}_{\boldsymbol{last}}^{\boldsymbol{enc}}$ denotes the target representation $\boldsymbol{v}$ and is used to be the hidden vector of the decoder at the first step, i.e. $\boldsymbol{h}_{\boldsymbol{0}}^{\boldsymbol{dec}} := \boldsymbol{h}_{\boldsymbol{last}}^{\boldsymbol{enc}}$. And EOS is the special token that signals the first step input of the decoding. Also, the decoding hidden representation $\boldsymbol{h}_{\boldsymbol{t}}^{\boldsymbol{dec}}$ is computed based on the distributed representation of the clean token $\boldsymbol{v}_{\mathcal{T}_{t-1}}$ and the hidden vector $\boldsymbol{h}_{\boldsymbol{t-1}}^{\boldsymbol{dec}}$ from the previous time step, i.e., $\boldsymbol{h}_{\boldsymbol{t}}^{\boldsymbol{dec}} := LSTM_{\theta'}^{dec}(\boldsymbol{v}_{\mathcal{T}_{t-1}}, \boldsymbol{h}_{\boldsymbol{t-1}}^{\boldsymbol{dec}})$. Note that LSTM is chosen as the computational unit in our model since some existing studies show that LSTM outperforms GRU in tasks requiring modeling long-distance relations [7].

Eventually, we predict $\boldsymbol{v}_{\mathcal{T}_t}^{pred}$ by the softmax function from the decoding hidden representation $\boldsymbol{h}_{\boldsymbol{t}}^{\boldsymbol{dec}}$ at each time step t,

$$\boldsymbol{v}_{\mathcal{T}_t}^{pred} := \frac{exp(\boldsymbol{W}^T \cdot \boldsymbol{h}_{\boldsymbol{t}}^{\boldsymbol{dec}} + \boldsymbol{b})}{\sum_{\boldsymbol{v} \in \mathcal{V}} exp(\boldsymbol{W}_{\boldsymbol{v}}^T \cdot \boldsymbol{h}_{\boldsymbol{t}}^{\boldsymbol{dec}} + b_{\boldsymbol{v}})}, \tag{4}$$

where $\boldsymbol{W} \in \mathbb{R}^{|h_t^{dec}| \times |\mathcal{V}|}$, $\boldsymbol{b} \in \mathbb{R}^{|\mathcal{V}|}$, $\boldsymbol{W}_{\boldsymbol{v}} \in \mathbb{R}^{|h_t^{dec}|}$ and $b_{\boldsymbol{v}} \in \mathbb{R}$ are the weights and bias represented by η, and softmax is the activation function. We define the loss function $\mathcal{L}(\boldsymbol{v}_{\mathcal{T}}, \boldsymbol{v}_{\mathcal{T}}^{pred})$ as the average sequence cross-entropy,

$$\mathcal{L}(\boldsymbol{v}_{\mathcal{T}}, \boldsymbol{v}_{\mathcal{T}}^{pred}) := \frac{1}{L} \cdot \sum_{i=1}^{L} \mathcal{H}\left(\boldsymbol{v}_{\mathcal{T}_i}, \boldsymbol{v}_{\tilde{\mathcal{T}}_i}^{pred}\right), \tag{5}$$

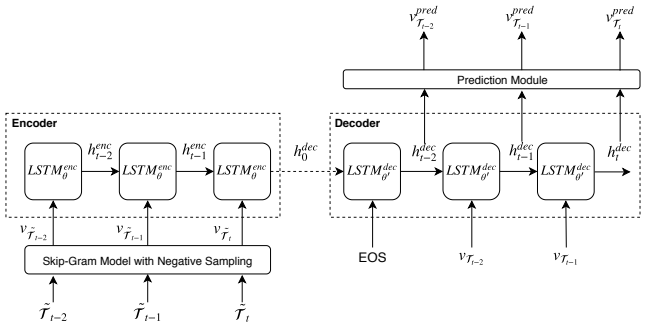


Figure 2: Denoising Sequential Encoder-Decoder Model. Take the sequence of the corrupted tokens $< \tilde{\mathcal{T}_{t-2}}, \tilde{\mathcal{T}_{t-1}}, \tilde{\mathcal{T}_t} >$ as an example, where EOS is a special token indicating the end of the input.

Algorithm 3 Denoising Sequential Encoder-Decoder (DSED) Model

Input: $\mathcal{D}$: the database of plays need to be embedded; d: the embedding dimension of each play; encoding function $LSTM_{\theta}^{enc}$ and decoding function $LSTM_{\theta'}^{dec}$; α: learning rate
Output: A trained DSED model

1: call Algorithm 1 to build the corpus $\mathcal{V}$;
2: **repeat**
3: get $< \mathcal{T}_1, \ldots, \mathcal{T}_L >$ from $\mathcal{V}$ for a play $\mathcal{P} \in \mathcal{D}$;
4: get corrupted version $< \tilde{\mathcal{T}}_1, \ldots, \tilde{\mathcal{T}}_L >$;
5: call Algorithm 2 to get distributed representations $\boldsymbol{v}_{\tilde{\mathcal{T}}}$ and $\boldsymbol{v}_{\mathcal{T}}$;
6: get $\boldsymbol{h}_{\boldsymbol{last}}^{\boldsymbol{enc}}$ (denoted as $\boldsymbol{v}$) and $\boldsymbol{h}_{\boldsymbol{last}}^{\boldsymbol{dec}}$ from $LSTM_{\theta}^{enc}$ and $LSTM_{\theta'}^{dec}$;
7: $(\theta, \theta', \eta) \leftarrow (\theta, \theta', \eta) - \alpha \nabla_{(\theta, \theta', \eta)} \mathcal{L}(\boldsymbol{v}_{\mathcal{T}}, \boldsymbol{v}_{\mathcal{T}}^{pred})$;
8: **until** No improvement on validation set

where $\mathcal{H}$ is the cross-entropy operator. Parameter θ of the encoding function $LSTM_{\theta}^{enc}$, θ' of the decoding function $LSTM_{\theta'}^{dec}$ and η are trained to minimize loss $\mathcal{L}(\boldsymbol{v}_{\mathcal{T}}, \boldsymbol{v}_{\mathcal{T}}^{pred})$ over a training set with the Adam stochastic gradient descent method.

Algorithm 3 presents the DSED model. It first calls Algorithm 1 to build the sports corpus $\mathcal{V}$, which contains segment tokens $\mathcal{T}$ (line 1). During the iterative training process (lines 2-8), it first maps the plays $\mathcal{P}$ to a sequence of segment tokens $< \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_L >$ of length L (line 3). Then, it constructs for each token $\mathcal{T}_i$ a corrupted version $\tilde{\mathcal{T}_i}$ (line 4) and call Algorithm 2 to get distributed representations $\boldsymbol{v}_{\mathcal{T}_i}$ and $\boldsymbol{v}_{\tilde{\mathcal{T}_i}}$ for $\mathcal{T}_i$ and $\tilde{\mathcal{T}_i}$, respectively (line 5). Next, it gets the reconstruction pairs $(\boldsymbol{v}_{\mathcal{T}}^{pred}, \boldsymbol{v}_{\mathcal{T}})$, which are computed by the encoder and decoder components of the learning model and uses an optimizer such as Adam stochastic gradient descent to optimize the parameters (lines 6-7).

4.4 Complexity Analysis

The time complexity for computing the similarity between two plays consists of two parts, namely one for learning the representations of the plays and the other for computing the distance between the learned representations in the form of vectors in the d-dimensional space. The former costs $O(n)$ time, where n is the length of the longest trajectory involved in the plays, and we explain as follows: (1) it takes $O(n)$ to convert a query play to a sequence of L segment matrices; (2) it takes $O(|\mathcal{V}|L)$ time to map the segment matrices to their corresponding segment tokens; (3) it takes $O(L)$ to map the segment tokens to their corresponding vectors; and (4)

Table 2: Dataset statistics.

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

it takes roughly $O(n)$ time to fed the vectors to the DSED model and obtains the target vector representation of the play. Since $|\boldsymbol{V}|$ could be regarded as a constant and L is bounded by n, we know this process takes $O(n)$ time. And the latter costs $O(d)$ which is obvious. Hence, the overall time complexity is $O(n + d)$.

5 EXPERIMENTS

5.1 Experimental Setup

Dataset. Our experiments are conducted on real-world soccer player tracking data [1]. The data consist of 7500 sequences and each sequence contains a segment of tracking data corresponding to actual game from a recent professional soccer league, totaling approximately 45 games worth of playing time and over 30 million data points, with redundant and "dead" situations removed. Each segment consists of the tracking data of three parts: 11 defense players, 11 attacking players and a ball. Each part contains (x, y) coordinates obtained at a sampling frequency of 10Hz. More specifically, the coordinates generally belong to the $[-52.5meters, +52.5meters]$ range along the x-axis, and $[-34meters, +34meters]$ range along the y-axis, with the very center of the pitch being $(0, 0)$. Table 2 presents the statistics of the dataset.

Baselines. To evaluate the effectiveness and efficiency of our deep representation learning method (called play2vec), we compare it with the following baseline similarity methods.

• DTW [29] and Frechet [1]. DTW and Frechet are two of the most widely adopted trajectory similarity measures in temporal sequence analyses. In particular, the Frechet is a metric-based distance, namely the distance is symmetric and satisfies the triangle inequality. However, the DTW distance is not a metric because it doesn't satisfy the triangle inequality. To measure the similarity between the two plays, we consider an agent-to-agent trajectory comparison method. We do this by first calculating the cost matrix between the trajectories of two plays based on the DTW and Frechet distance, respectively and then computing the optimal assignment using the Hungarian algorithm [12].

• Chalkboard [24]. This method is the state-of-the-art method proposed for sports play retrieval and overcomes the exhaustive comparing problem of an agent-to-agent method by using a role-based representation to enable fast alignment of trajectories. Additionally, effective templating and hashing techniques are employed to support users' queries at interactive speeds.

• EMDT [11]. EMDT is proposed to study the similarity of trajectory sets over the road network and defines a novel similarity measure by borrowing the idea of the Earth Mover's Distance. However, their problem scenario is on the road network and the implementation of EMDT partly requires an map matching algorithm. We adapt this method by regarding the center point of each cell as a node on a road network and mapping the sample points of all trajectories to their nearest center points.

Parameter Setting. The default size of cells is 3 meters and short duration of segments is 1 second in the experiments. After building a sports corpus via the Jaccard index (the threshold is 0.3), we got 50,465 unique tokens for our dataset. We use a 2-layer LSTM as the computational unit in LSTM-Encoder. The representation dimension of the learned segment tokens and plays are set to 20 and 50 respectively. The context window size in the distributed representation learning is set to 5. The α-smoother is set to 3/4 following the negative sampling in word2vec. Additionally, in the training process, we train our model on 5k generated plays with randomly noise and dropping rate and adopt Adam stochastic gradient descent with an initial learning rate of 0.01. In order to avoid the gradient vanishing problem, a maximum gradient norm constraint is used and set to 5. For the parameters of baselines, we follow their strategies described in the original papers.

Evaluation Platform. All the methods are implemented in Python 3.6. The implementation of play2vec is based on tensorflow 1.8.0, which is available at https://github.com/zhengwang125/play2vec/. The experiments are conducted on a machine with Intel(R) Xeon(R) CPU E5-1620 v2 @3.70GHz 16.0GB RAM and one Nvidia GeForce GTX 1070 GPU.

5.2 Effectiveness Evaluation

Overall effectiveness. We first study the effectiveness of play2vec. The lack of ground-truth makes it a challenging problem to evaluate the accuracy. To overcome it, we follow three recent studies [14, 21, 26] which propose to quantify the accuracy of trajectory similarity with Self-similarity, Cross-similarity and KNN-similarity comparisons, respectively. There are two frequently used parameters: noise rate (radius is set to 1 meter.) and dropping rate with varying values from 0.2 to 0.6. The two parameters are to measure the probabilities of adding noise or dropping sampling points of each trajectory in a play, respectively.

5.2.1 Self-similarity Comparison. In this experiment, we randomly choose 50 plays to form the query set (denoted as Q) and 1000 plays as the target database (denoted as D) from the dataset. For each play $P \in Q$, we create two sub-plays by randomly sampling 20% points for each trajectory in the play, denoted as P_a and P_b, and we use them to construct two datasets $Q_a = \{P_a\}$ and $Q_b = \{P_b\}$. Similarly, we get D_a and D_b from the target database D. Then for each query $P_a \in Q_a$, we compute the rank of P_b in the database $Q_b \cup D_b$ using different methods. Ideally, P_b should be ranked at the top since P_a and P_b are generated from the same play P. To evaluate the robustness of different approaches to noise, we consider introducing two types of noises. First, we corrupt each trajectory of each play in both Q_a and $Q_b \cup D_b$ as follows: We randomly sample a fraction of the points (denoted by noise rate r_1) and for each sample point we distort the coordinate values by adding Gaussian noises with a standard normal distribution. We vary r_1 from 0.2 to 0.6 and report the mean rank of the queries in Table 3. Second, we randomly drop a fraction of points from each

[1]Data Source: STATS, copyright 2019 (https://www.stats.com/artificial-intelligence)

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**

Table 4: Self-similarity of varying dropping rate.

dropping rate	0.2	0.3	0.4	0.5	0.6
DTW	115.60	118.20	124.80	129.00	130.80
Frechet	144.60	155.20	176.80	185.60	202.00
Chalkboard	57.80	60.40	62.40	68.20	69.60
EMDT	56.20	70.40	70.60	91.40	96.40
play2vec	**26.24**	**29.50**	**39.74**	**50.20**	**54.00**

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**

Table 6: Cross-similarity of varying dropping rate.

dropping rate	0.2	0.3	0.4	0.5	0.6
DTW	0.198	0.290	0.397	0.500	0.588
Frechet	0.251	0.253	0.254	0.256	0.257
Chalkboard	0.197	0.299	0.397	0.490	0.600
EMDT	0.295	0.302	0.303	0.304	0.322
play2vec	**0.002**	**0.008**	**0.009**	**0.017**	**0.032**

trajectory of each play in both Q_a and $Q_b \cup D_b$. We vary dropping rate r_2 from 0.2 to 0.6 and report the mean rank of the queries in Table 4. Note that the mean rank in self-similarity evaluation is a rank-based metric defined as $\frac{1}{|Q_a|}\sum_{P_a} rank(P_b)$, where $rank(P_b)$ denotes the rank of P_b in $Q_b \cup D_b$ for a query $P_a \in Q_a$. We observe that play2vec consistently outperforms the other methods by a large margin as we vary the two types of noise. We also observe that most of the methods are not very sensitive to the noise rate except that DTW and EMDT degrade quickly when we increase the noise rate to 0.5. This is because the matching cost of DTW is determined by the pairwise point-matching and errors will be accumulated with noises. However, Frechet maintains an infimum of the matching cost that is robust to noise changes. With regard to Chalkboard, it splits trajectories into overlapping segments, which can alleviate the noise interruption to some degree. Moreover, with the increase of the noise rate, the gap between EMDT and other methods grows and the mean rank of EMDT is consistently very large. This is because after adding noise, the cell centers corresponding to P_a and P_b are very different, even if they are sampled from the same play. When we vary the dropping rate, the mean ranks of DTW and Frechet are significantly larger than other methods. This implies that the pairwise point-matching methods based on agent-to-agent alignment may not be able to handle the case when sampling rate is low.

5.2.2 Cross-similarity Comparison. A good similarity measure should preserve the distance between two plays regardless of the sampling rate or noise interference. We use a metric from the literatures [14, 26] to evaluate the effectiveness for Cross-similarity comparison, namely Cross Distance Deviation (CDD) as defined below.

$$CDD(P_a, P_b) = \frac{|S(P_{a'}(r), P_{b'}(r)) - S(P_a, P_b)|}{S(P_a, P_b)}, \tag{6}$$

where $S(\cdot,\cdot)$ is a similarity measure such as DTW or Frechet; P_a and P_b are two original plays; $P_{a'}(r)$ and $P_{b'}(r)$ are their variants that are obtained by randomly dropping points (or adding noise) with rate r. A small CDD value indicates that an algorithm is robust and is able to preserve the original distance well. In this experiment, we randomly select 1,000 play pairs (P_a, P_b) from the dataset. The average CDD results are reported in Table 5 and Table 6. We observe that play2vec outperforms other baselines consistently for different noise and dropping rates. Note that play2vec is very close to the ground truth over various dropping rates because a cell of the grid maps is considered occupied only if one sample point falls in the cell. Therefore, play2vec can effectively handle the low sampling issue of data.

5.2.3 KNN-similarity Comparison. In this experiment, we study the accuracy and robustness of play2vec and the other baselines for KNN-similarity search on plays when we vary the dropping rate or noise rate. To circumvent the issue of lack of ground-truth, we follow the experimental methodology that is proposed by existing studies [14, 21]: We first randomly select 20 plays as the query set and 500 plays as the target database, and for each query we employ each method to find its Top-K plays as the ground-truth of each method; Then we corrupt each play in the target database by randomly dropping points or adding noise, and retrieve the Top-K plays from the corrupted database; Finally, we compare the retrieved Top-K plays against the ground-truth to compute the precision, i.e., the proportion of true Top-K plays among the retrieved Top-K plays. We vary the value of K by 20, 30, 40, and vary the dropping rate or noise rate from 0.2 to 0.6. The average precision results are reported in Figure 3. We observe that play2vec performs the best consistently. As expected, the precision of all methods decreases when the noise or dropping rate increases. We observe similar trends for all methods over different K values. The precision of DTW drops rapidly when the dropping rate is more than 0.5. EMDT performs the worst when we inject noise into the target database due to the same reason we analyzed for the Self-similarity comparison experiment. Additionally, Frechet is more robust than other baselines when we corrupt the target database by dropping points.

5.2.4 Parameter Study. We next evaluate the effect of the cell size on the effectiveness of play2vec. Intuitively, a small cell size provides a higher resolution of the sports scene, but it also generates more tokens, which lead to higher training complexity and reduce

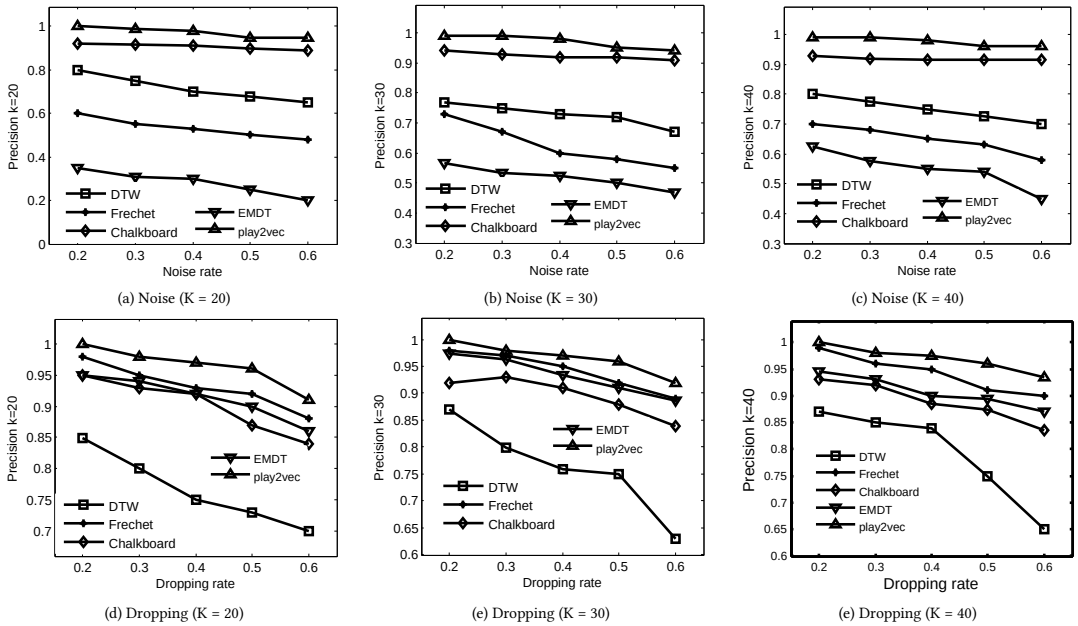


Figure 3: KNN results when varying the noise and dropping rate from 0.2 to 0.6 for K = 20, 30, 40.

robustness. In Table 7, we report the performance in answering Self-similarity, Cross-similarity and KNN-similarity, where r_n and r_d are noise rate and dropping rate, respectively. We observe that the performance becomes better as the cell size grows from 1m to 3m, and drops for Cross-similarity and KNN similarity when the cell size becomes 4m. With the smallest cell size (1m) play2vec performs the worst. This is probably because the high model complexity makes it difficult to train and this is in line with our intuition. Therefore, we set the cell size at 3 meters for the other experiments because it offers a better robustness.

5.3 Efficiency Evaluation

This set of experiments is to evaluate the efficiency of different methods for the sports play retrieval. Figure 4 shows the average cost of computing the similarity between a query play and the plays when we vary the size of the target database. Note that the y-axis is in logarithmic scale. Clearly, EMDT performs extremely slow because the overall time cost of the Earth Mover's Distance is super-cubic to the number of the cells over a soccer pitch [22]. DTW and Frechet have similar running time because they are pair-wise point-matching methods, which has quadratic computational complexity. Chalkboard is the most efficient algorithm among the baselines because it adopts a fast role-based representation to avoid exhaustively comparing computation in agent-to-agent alignment. Note that in Chalkboard, we perform offline preprocessing and do not count the time for fair comparison. play2vec performs the best among all methods and is over an order of magnitude faster than the most efficient baseline Chalkboard. This is because play2vec takes linear time to retrieve similar plays as discussed in Section 4.4. We also notice that play2vec scales linearly with the database size and the disparity between them increases as the size of the target database grows.

5.4 User Study

Since Chalkboard performs the best among all the baselines, and is dedicated for similar play retrieval, we compare play2vec with Chalkboard for play retrieval via a user study. We randomly select 10 plays as the query set and for each query we employ play2vec and Chalkboard to retrieve Top-1 play from the target database, respectively. We recruited seven volunteers with strong soccer background to annotate the relevance of retrieved plays. We first spent 10 minutes to get everyone understand the images in Figure 6. Then, for each of the 10 queries each volunteer specifies the most relevant result between the Top-1 result retrieved by play2vec and the Top-1 result retrieved by Chalkboard. Note that volunteers do not know which result is from which method. We show the scores of the 10 queries for both methods in Figure 5. We observe that play2vec performs much better than Chalkboard for 8 out of the 10 queries. Overall, play2vec gets 82.86% votes (over 70 votes) while Chalkboard only gets 7.14% votes. We illustrate the results by showing the Top-5 results of play2vec and Chalkboard, respectively, on query Q1 in Figure 7. We observe that Q1 is a classical "lofted pass" tactics in a soccer match. We find that the Top-1 result of play2vec matches the query very well and Top 2-5 results also maintain a high consistency between results and the query. However, Chalkboard fails to return relevant results, which is due to the imperfect alignment as discussed in Section 1.

6 CONCLUSION

In this paper, we study the problem of sports play retrieval and present the first deep learning based method for computing the similarity between two sports plays. Inspired by the success of modeling word similarity, we extend the Skip-Gram with Negative Sampling (SGNS) model and develop a new Denoising Sequential

Table 7: The influence of the cell size.

cell size	1m			2m			3m			4m		
#tokens	87382			67681			50465			22570		
evaluation	Self	Cross	KNN	Self	Cross	KNN	Self	Cross	KNN	Self	Cross	KNN
$r_n = 0.5$	80.67	0.248	0.68	30.60	0.178	0.87	25.30	**0.093**	**0.95**	**25.00**	0.101	0.94
$r_d = 0.5$	72.50	0.198	0.70	69.40	0.087	0.90	54.00	**0.017**	**0.96**	**53.20**	0.022	0.94

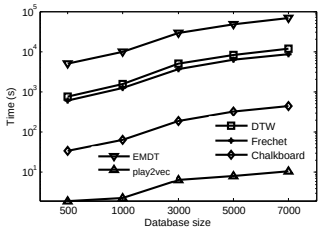


Figure 4: Efficiency.

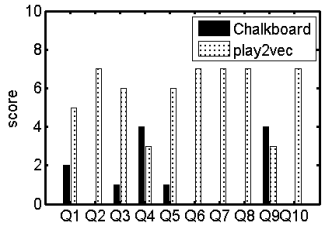


Figure 5: User Study.

Encoder-Decoder (DSED) framework to learn consistent representations. Our solution is robust to the non-uniform sampling rates and noises and only takes a linear time to compute the similarity. Also, we evaluate experiments on real-world soccer match data and find that it achieves better effectiveness and efficiency than the existing methods. In the future, we plan to develop indexing techniques like Locality-Sensitive Hashing to further accelerate the retrieval for large-scale datasets.

Acknowledgments. This work is supported in part by a MOE Tier-2 grant MOE2016-T2-1-137, a MOE Tier-1 grant RG31/17 and NTU SUG grant M4082302.020. The authors would also like to thank Tobias Emrich, Matthias Schubert, and Deepak Padmanabhan for some initial discussions.

REFERENCES

[1] Helmut Alt and Michael Godau. 1995. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications* 5, 01n02 (1995), 75–91.

[2] Raquel Aoki, Renato M Assuncao, and Pedro OS Vaz de Melo. 2017. Luck is hard to beat: The difficulty of sports prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1367–1376.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[4] Alina Bialkowski, Patrick Lucey, Peter Carr, Yisong Yue, Sridha Sridharan, and Iain Matthews. 2014. Large-scale analysis of soccer matches using spatiotemporal tracking data. In *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 725–730.

[5] Lei Chen and Raymond Ng. 2004. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 792–803.

[6] Lei Chen, M Tamer Özsu, and Vincent Oria. 2005. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 491–502.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[8] Tom Decroos, Jan Van Haaren, and Jesse Davis. 2018. Automatic Discovery of Tactics in Spatio-Temporal Soccer Match Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 223–232.

[9] Mingyang Di, Diego Klabjan, Long Sha, and Patrick Lucey. 2018. Large-Scale Adversarial Sports Play Retrieval with Learning to Rank. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 12, 6 (2018), 69.

[10] Joachim Gudmundsson and Michael Horton. 2017. Spatio-temporal analysis of team sports. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 22.

[11] Dan He, Boyu Ruan, Bolong Zheng, and Xiaofang Zhou. 2018. *Trajectory Set Similarity Measure: An EMD-Based Approach*. 28–40. https://doi.org/10.1007/978-3-319-92013-9_3

[12] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[13] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. 1188–1196.

[14] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 617–628.

[15] Tie-Yan Liu, Wei-Ying Ma, and Hong-Jiang Zhang. 2005. Effective feature extraction for play detection in american football video. In *Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International*. IEEE, 164–171.

[16] Patrick Lucey, Alina Bialkowski, Peter Carr, Stuart Morgan, Iain Matthews, and Yaser Sheikh. 2013. Representing and discovering adversarial team behaviors using player roles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2706–2713.

[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[18] Andrew Miller, Luke Bornn, Ryan Adams, and Kirk Goldsberry. 2014. Factorized point process intensities: A spatial analysis of professional basketball. In *International Conference on Machine Learning*. 235–243.

[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[20] Lukas Probst, Ihab Al Kabary, Rufus Lobo, Fabian Rauschenbach, Heiko Schuldt, Philipp Seidenschwarz, and Martin Rumo. 2018. SportSense: User Interface for Sketch-Based Spatio-Temporal Team Sports Video Scene Retrieval. In *Proceedings of the first workshop on User Interface for Spatial and Temporal Data Analysis (UISTDA'18). CEUR-WS.*

[21] Sayan Ranu, P Deepak, Aditya D Telang, Prasad Deshpande, Sriram Raghavan, et al. 2015. Indexing and matching trajectories under inconsistent sampling rates. In *2015 IEEE 31st International Conference on Data Engineering (ICDE)*. IEEE, 999–1010.

[22] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision* 40, 2 (2000), 99–121.

[23] Haşim Sak, Andrew Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association.*

[24] Long Sha, Patrick Lucey, Yisong Yue, Peter Carr, Charlie Rohlf, and Iain Matthews. 2016. Chalkboarding: A new spatiotemporal query paradigm for sports play retrieval. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*. ACM, 336–347.

[25] Long Sha, Patrick Lucey, Yisong Yue, Xinyu Wei, Jennifer Hobbs, Charlie Rohlf, and Sridha Sridharan. 2018. Interactive Sports Analytics: An Intelligent Interface for Utilizing Trajectories for Interactive Sports Play Retrieval and Analytics. *ACM Transactions on Computer-Human Interaction (TOCHI)* 25, 2 (2018), 13.

[26] Han Su, Kai Zheng, Haozhou Wang, Jiamin Huang, and Xiaofang Zhou. 2013. Calibrating trajectory data for similarity-based analysis. In *Proceedings of the 2013 ACM SIGMOD international conference on management of data*. ACM, 833–844.

[27] Zheng Wang, Ce Ju, Gao Cong, and Cheng Long. 2018. Representation Learning for Spatial Graphs. *arXiv preprint arXiv:1812.06668* (2018).

[28] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. Computing Trajectory Similarity in Linear Time: A Generic Seed-Guided Neural Metric Learning Approach. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE.

[29] Byoung-Kee Yi, HV Jagadish, and Christos Faloutsos. 1998. Efficient retrieval of similar time sequences under time warping. In *Data Engineering, 1998. Proceedings., 14th International Conference on*. IEEE, 201–208.

[30] Yisong Yue, Patrick Lucey, Peter Carr, Alina Bialkowski, and Iain Matthews. 2014. Learning fine-grained spatial models for dynamic sports play prediction. In *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 670–679.

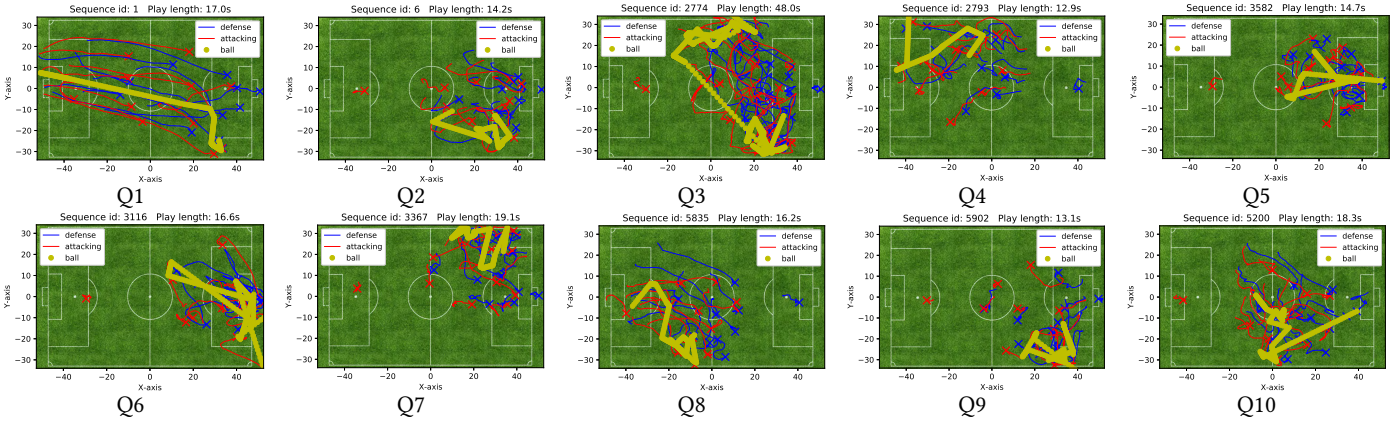


Figure 6: Ten queries which are used for user study. These trajectories cover a wide range of zones in the pitch. The small "x" is the end point of the movement.

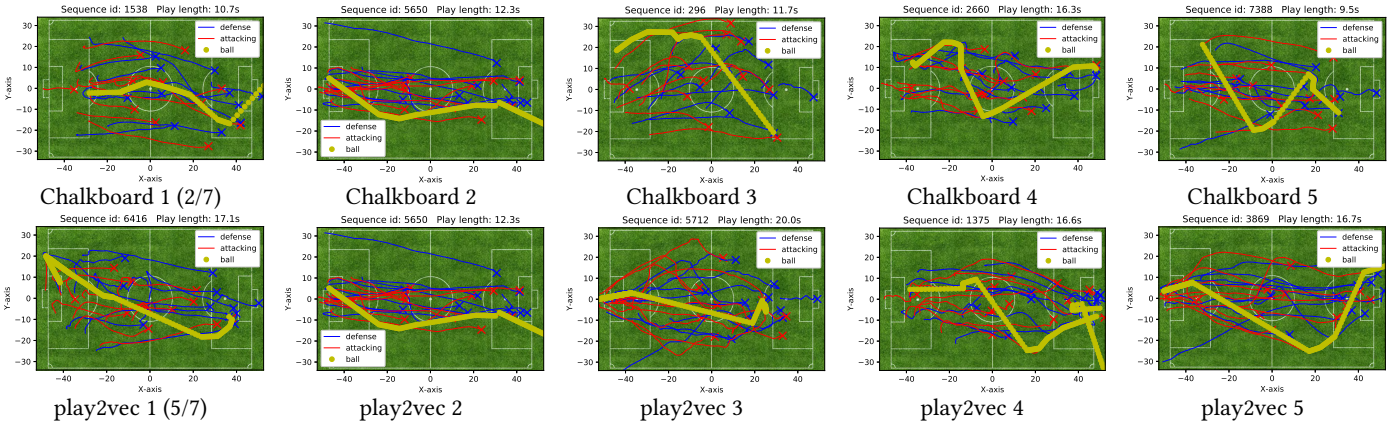


Figure 7: A retrieval example for Q1 of using two methods. The Top-5 results (from left to right) returned by the Chalkboard and play2vec. (2/7) means the 2 participants support this result in 7 participants.

SUPPLEMENTAL

Here, we show that the queries for user study in Figures 6 and Top-5 results of query Q1 are used for case study in Figure 7. More details are described in subsection 5.4. Figure 8 and Figure 9 show the Top-1 results of the Chalkboard and play2vec for Q2 to Q10.

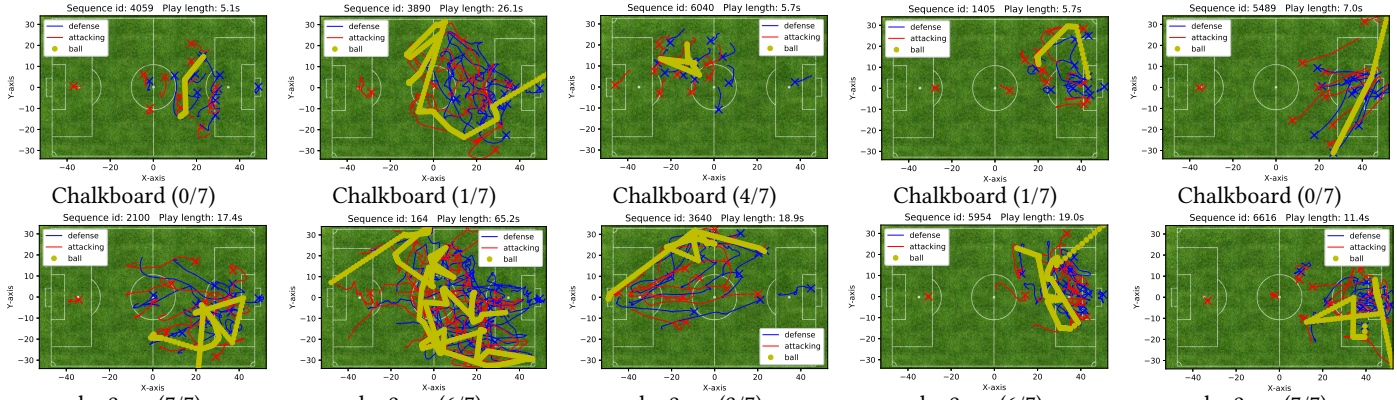


Figure 8: Top-1 results for Q2-Q6 (from left to right).

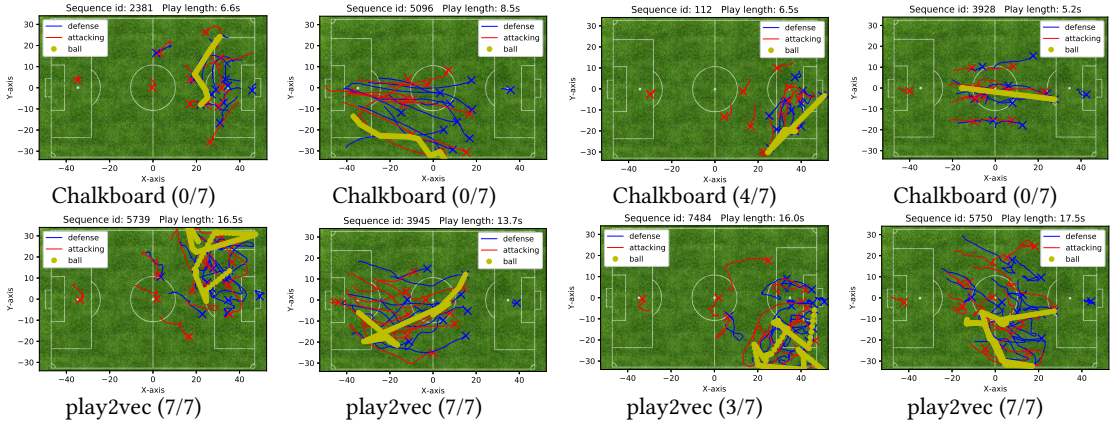


Figure 9: Top-1 results for Q7-Q10 (from left to right).