Time-Aware User Embeddings as a Service

Martin Pavlovski*
Temple University
Philadelphia, PA
martin.pavlovski@temple.edu

Jelena Gligorijevic*
Yahoo! Research
Sunnyvale, CA
jelenas@verizonmedia.com

Ivan Stojkovic
Yahoo! Research
Sunnyvale, CA
ivans@verizonmedia.com

Shubham Agrawal
Yahoo! Research
Sunnyvale, CA
shubhama@verizonmedia.com

Shabhareesh Komirishetty
Yahoo! Research
Sunnyvale, CA
shabha@verizonmedia.com

Djordje Gligorijevic
Yahoo! Research
Sunnyvale, CA
djordje@verizonmedia.com

Narayan Bhamidipati
Yahoo! Research
Sunnyvale, CA
narayanb@verizonmedia.com

Zoran Obradovic
Temple University
Philadelphia, PA
zoran.obradovic@temple.edu

ABSTRACT

Digital media companies typically collect rich data in the form of sequences of online user activities. Such data is used in various applications, involving tasks ranging from click or conversion prediction to recommendation or user segmentation. Nonetheless, each application depends upon specialized feature engineering that requires a lot of effort and typically disregards the time-varying nature of the online user behavior. Learning time-preserving vector representations of users (user embeddings), irrespective of a specific task, would save redundant effort and potentially lead to higher embedding quality. To that end, we address the limitations of the current state-of-the-art self-supervised methods for task-independent (unsupervised) sequence embedding, and propose a novel Time-Aware Sequential Autoencoder (TASA) that accounts for the temporal aspects of sequences of activities. The generated embeddings are intended to be readily accessible for many problem formulations and seamlessly applicable to desired tasks, thus sidestepping the burden of task-driven feature engineering. The proposed TASA shows improvements over alternative self-supervised models in terms of sequence reconstruction. Moreover, the embeddings generated by TASA yield increases in predictive performance on both proprietary and public data. It also achieves comparable results to supervised approaches that are trained on individual tasks separately and require substantially more computational effort. TASA has been incorporated within a pipeline designed to provide *time-aware user embeddings as a service*, and the use of its embeddings exhibited lifts in conversion prediction AUC on four audiences.

*Both authors contributed equally to this research.

KDD '20, August 23–27, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403371

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; **Neural networks**; Dimensionality reduction and manifold learning; • **Information systems** → *Online advertising*; *Computational advertising*.

KEYWORDS

user representation; neural embeddings; sequential models

ACM Reference Format:
Martin Pavlovski, Jelena Gligorijevic, Ivan Stojkovic, Shubham Agrawal, Shabhareesh Komirishetty, Djordje Gligorijevic, Narayan Bhamidipati, and Zoran Obradovic. 2020. Time-Aware User Embeddings as a Service. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining USB Stick (KDD '20), August 23–27, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403371

1 INTRODUCTION

The present-era Internet businesses operate on the basis of immense amounts of user-generated data. Digital media companies have access to such data sources and seek to leverage them for understanding the online behavior of users. Nevertheless, modeling user behavior is typically tied to various prediction problems in the realm of advertising, user profiling, or recommendation. Therefore, individual teams in digital media companies are working on different user-related prediction problems. Often, a prediction problem needs to be addressed for various clients/advertisers/campaigns individually, thus introducing a separate prediction task for each of them. Considering the broad range of such prediction tasks, addressing them requires investing significant time and effort into feature engineering. Even though different teams may be working on different tasks, if the tasks are centered around users, the teams might be using overlapping aspects of the same data source(s). This results in redundant preprocessing efforts for the teams involved. To avoid feature engineering altogether, a team might consider supervised deep learning models to learn latent representations of users tailored for the team's particular task without manually preprocessing the original user data. However, these models are usually heavily parameterized in accordance with the large scale

of the data. Since a separate deep model would need to be trained for each task, this is prohibitively expensive keeping in mind that low serving time remains among the top priorities. On the other hand, low latency can be maintained to accommodate serving time. For this purpose, lighter models may be leveraged. Despite their efficiency, the main drawback of these models is that they require the input to be properly encoded (usually through one-hot encoding) which results in the creation of billions of sparse features. For instance, consider a Logistic Regression model that learns to predict user conversions based on a sequence of activities that the user performed. Due to the large number of possible web activities that a user can perform, the number of resulting engineered features (e.g., one-hot encoded, including cross-features) can be in the order of billions. Moreover, having to consider different web activities for different tasks requires redundant construction of a huge number of features for each task which introduces a notable computational burden to the individual teams.

An alternative to feature engineering is to use solely a set of task-invariant user features such as demographic features (e.g. location, age and gender), which are usually accessible to all (or most of the teams) in digital media companies. These features are commonly used for various tasks due to their low cardinality.

Motivated by the aforementioned challenges and the recent advancements in unsupervised representation learning [5, 15, 32–34, 38], we propose the Time-Aware Sequential Autoencoder (TASA), an autoencoder model that learns time-preserving *representations of user trails* (also referred to as *user embeddings*) from a collection of online activity sequences. The fact that TASA (1) embeds users in an *unsupervised* manner and (2) considers all activities that a user may perform instead of focusing on a subset of activities tailored for a certain task, allows for learning task-independent representations. Moreover, TASA learns additional temporal scores for each activity to account for the irregular time gaps between consecutive activities and preserves this information in the resulting user embeddings. For the advantages of TASA over other autoencoder-based embedding approaches, the reader is referred to Table 1.

Further, we propose a pipeline that integrates TASA to allow for using *embeddings as a service* which in this work is described through, but is not limited to, user embeddings. As a part of the pipeline, the user activities collected during a certain period of time are fetched from multiple data sources and organized into sequences, also referred to as *user trails*. TASA is then trained on the constructed user trails. Upon training, TASA outputs the embeddings for all users and stores them in a centralized database as M-dimensional continuous vectors, where M is typically in the order of hundreds. Analogously, the learned parameters of TASA are stored in a separate database. From that point onward, any incoming user trail, or individual activity, can be embedded using the trained TASA model and its embedding can be stored in the centralized embedding database. Indeed, one of the pipeline's merits is its ability to generate embeddings even for entirely new, previously unobserved users as long as TASA has observed some of the activities from their trails.

In the proposed pipeline, at all times, different teams are able to (1) query the centralized database, (2) obtain a set of embeddings, (3) concatenate the retrieved M-dimensional embeddings to the features that are already in use by the teams and (4) directly continue using low-latency supervised models for their downstream prediction tasks, without the need of any feature engineering or manual preprocessing interventions. Constituting a central component of the pipeline, the time-aware autoencoder design of TASA enables automatic generation of low-dimensional, time-preserving user embeddings applicable to any user-level task while maintaining the ability to use low-latency supervised models. We believe that providing these embeddings as a service to the teams will sidestep the burden of task-driven feature engineering.

In the conducted experiments, TASA was run on both proprietary and public data and compared against a number of autoencoder variants, some of which constitute the current state-of-the-art in unsupervised representation learning. The performance of TASA and its alternatives has been assessed in both unsupervised and supervised settings. The experimental results indicate that autoencoders capable of learning time-preserving embeddings lead to more accurate sequence reconstruction. Subsequently, the performance of supervised baselines was evaluated with respect to several supervised tasks. The supervised models for these tasks were trained also on the user embeddings learned by each unsupervised embedding approach, in addition to commonly available information about users. The findings suggest that using TASA's user embeddings yields 1) the largest percentage increases in AUC consistently for all supervised tasks compared to the alternative unsupervised embedding variants, and 2) comparable results to a supervised high-latency deep model learned on each task separately. TASA also outperforms a low-latency logistic regression model, learned on each task separately with one-hot encoded features in addition to commonly available demographic information about users. Note that, even though it is out of the scope of this paper, learning time-preserving user representations, irrespective of a certain task, can be applied beyond the realm of supervised tasks. Indeed, the generated user embeddings can also be used for different tasks of unsupervised nature, including user co-clustering and building custom user segments, among others.

TASA was also integrated into the pipeline designed to provide *time-aware user embeddings as a service*, which is currently deployed as an internal tool and is being utilized by several teams in the company as a source of additional user features, mainly for offline experimentation. Offline conversion prediction experiments have been conducted on four different audiences. The findings provide evidence that improvements in AUC are achieved when the TASA-generated user embeddings, within the service pipeline, are leveraged in addition to the current features used in production.

2 RELATED WORK

Unsupervised representation learning as an umbrella term encompassing multiple concepts, mainly dimensionality reduction, feature projection and manifold learning, has been extensively studied. A large spectrum of unsupervised embedding methods have been developed, from learning clustering-based representations, up to linear [19, 23, 28] and non-linear [16, 29, 35, 39] embedding methods, intended mostly for dimensionality reduction and manifold learning. Recently, end-to-end approaches for learning compact yet informative feature representations gained in popularity. Autoencoders constitute a representative class of such approaches.

Table 1: Advantages and limitations of the proposed TASA and alternative autoencoder variants. More details on the alternative approaches are provided in Section 5.1.1.

Model	Dense vectorial embeddings	Sequential information	Temporal information	Leverages stop features as temporal timestamps	Scores the influences of activities and timestamps on the user embeddings
Conventional AE	✓	✗	✗	✗	✗
seq2seq [38]	✓	✓	✗	✗	✗
TA-seq2seq [5]	✓	✓	✓	✗	✗
ISA [32]	✓	✓	✓	✓	✗
TASA (proposed)	✓	✓	✓	✓	✓

Initially proposed for unsupervised pre-training [4], a large body of literature on autoencoders spanning over three decades focuses on unsupervised feature learning [7, 17, 22].

A broad range of autoencoder flavors have been introduced over the years to handle data of different type and nature. For instance, one line of autoencoders was adapted to learn representations of sequential inputs. For this purpose, neural models that capture the temporal dynamics of sequences, such as Recurrent Neural Networks (RNNs), were employed to learn sequence representations. Typically, in a sequential autoencoding framework an RNN encodes a set of sequences into fixed-length vectorial representations, followed by another RNN that decodes these representations back to the original sequences. The idea stems from the concept of sequence-to-sequence learning and the introduction of the seq2seq model [38] for language translation. The initial seq2seq model utilized Long Short-Term Memory (LSTM) networks to encode English sentences and reconstruct them into their French translations. Seq2seq autoencoders [1, 9, 40], on the other hand, aim at reconstructing the same input sequences from their own representations to which they are encoded. Nevertheless, sequential autoencoders learn representations that preserve the sequential order within sequences assuming constant elapsed times between their constituent elements. To account for irregular time gaps between consecutive sequence elements, a Time-Aware LSTM (T-LSTM) autoencoder was proposed in [5, 34]. Recently, T-LSTM found applications in interpretable representation learning for disease progression modeling [3, 41] and user conversion prediction in prospective advertising [13]. Another recent study [32] also leveraged the concept of capturing irregular time gaps by introducing *stop features* to serve as temporal stamps in a sequential autoencoding framework. The reconstruction process of the framework integrated two classical mechanisms in order to account for both (1) the global silhouette information that underlies a sequence and (2) the local temporal dependencies among the sequence constituents. This allowed for learning to differentiate speakers given their speech sequence samples in addition to recognizing solely the text they utter.

In general, autoencoders have shown to be effective in generating compact representations for subsequent supervised learning tasks in a broad range of domains including information retrieval [6, 21, 33, 36], natural language processing [2, 10, 11, 24, 26], computer vision [31, 37], audio signal processing [1, 9], text-to-speech synthesis [40], multi-task learning [27], network embedding [15], among others. For a more thorough overview of autoencoders and their applications, we refer the reader to [14, Ch. 14].

3 TIME-AWARE SEQUENTIAL AUTOENCODER (TASA)

The proposed model, TASA, is an autoencoder variant that learns time-preserving representations in an unsupervised manner. The input to TASA is a sequence of activities $\{a_1, \ldots, a_L, a_{L+1}\}$, along with their corresponding timestamps $\{t_1, \ldots, t_L, t_{L+1}\}$. Note that the actual input sequence is of variable length L, whereas $a_{L+1} = a_{EOS}$ is an end-of-sequence token that allows for handling sequences of different lengths. Correspondingly, t_{L+1} is set to be equal to the timestamp of the most recent activity a_L. TASA first encodes the entire sequence into a fixed-length vector representation $\mathbf{h}$ that reflects the sequential and temporal dependencies among the activities. Thereafter, a sequence of activities is decoded from the learned representation, as similar as possible to the input sequence. By leveraging this principle, TASA enforces learning of sequence representations that prioritize informative activity properties, while preserving the sequential and temporal dependencies among the activities. The following contains a detailed description of TASA's building blocks.

3.1 Activity Embedding

Initially, every unique activity $a \in V$ is assigned an embedding $\mathbf{v} \in \mathbb{R}^D$, given that V is a vocabulary containing all distinct activities, i.e. $V = \{a^{(1)}, \ldots, a^{(|V|)}\}$ such that $a^{(j)} \neq a^{(k)}, j \neq k$. Note that two additional tokens are reserved in V for the *start* and *end-of-sequence* tokens a_{start} and a_{EOS}, respectively.

The activity embeddings are initialized to random uniform values and a function $g : V \rightarrow \mathbb{R}^D$ is used to map each activity a_j to its corresponding embedding $\mathbf{v}_j = g(a_j)$, for every $j = 1, \ldots, L+1$. The embeddings $\mathbf{v}_j$ are then organized into a sequence $\{\mathbf{v}_1, \ldots, \mathbf{v}_L, \mathbf{v}_{L+1}\}$ in the same order as their corresponding activities in the input sequence $\{a_1, \ldots, a_L, a_{L+1}\}$.

3.2 Temporal Score Learning

To capture the irregular time intervals between consecutive activities, an additional feature $\tau_j = \frac{t_j}{t_{L+1}}$ is created for a_j, where t_j is the timestamp of a_j. Following the terminology from [32], the resulting features $\{\tau_1, \ldots, \tau_L, \tau_{L+1}\}$ are referred to as *stop features*. The intuition behind a stop feature suggests that it can act as an activity's relative "closeness", in time, to the end of a sequence. Increasing linearly with the activities' timestamps, the closer an activity occurred to the latest timestamp in a sequence, the closer its stop feature value is to 1. In addition to the stop features, each activity a_j is

mapped to a pair of latent parameters $\theta_j, \mu_j \in \mathbb{R}$. Similarly to [3, 13], θ_j serves to model the influence of activities to the sequence embeddings. On the other hand, μ_j and τ_j model the influence that activity occurrence times have on the sequence embeddings. Through the reconstruction process, the learned embeddings should preserve these influences through the latent parameters. For this purpose, a temporal score is defined for each a_j as

$$\delta_j = \sigma(\theta_j + \mu_j \tau_j), \tag{1}$$

where σ denotes a sigmoid (logistic) activation function. Here, θ_j is intended to measure the initial influence of a_j. On the other hand, μ_j acts as the change in the influence based on a_j's "recency". The extent to which the effect of a_j changes through time depends on the magnitude of μ_j. However, $\theta_j + \mu_j \tau_j$ is passed through a sigmoid activation function to represent a probability. Hence, for a given θ_j, large positive and low negative values of μ_j will push δ_j closer to 0 and 1, respectively. As suggested in [13], a sigmoid activation is proposed for activity-specific modeling of the influence as opposed to a softmax activation which enforces the influences of all activities to sum up to 1. This accounts for cases where the same activity occurred in multiple instances within the same sequence, by putting more attention to more recent occurrences. Finally, the temporal scores are used to scale the activity embeddings:

$$\hat{\mathbf{v}}_j = \delta_j \mathbf{v}_j. \tag{2}$$

3.3 Sequential Encoding

Once the temporally scored activity embeddings are generated, the sequences of activity embeddings are further encoded into M-dimensional representations. To that end, a long short-term memory (LSTM) network is utilized. The LSTM network takes the sequence of temporally scored activity embeddings $\{\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_L, \hat{\mathbf{v}}_{L+1}\}$ and passes each $\hat{\mathbf{v}}_j$ through a forget gate $\mathbf{f}_j$, an input gate $\mathbf{i}_j$, an output gate $\mathbf{o}_j$ and a cell $\mathbf{C}_j$:

$$\mathbf{f}_j = \sigma(\mathbf{W}_f \hat{\mathbf{v}}_j + \mathbf{U}_f \mathbf{h}_{j-1} + \mathbf{b}_f), \tag{3}$$

$$\mathbf{i}_j = \sigma(\mathbf{W}_i \hat{\mathbf{v}}_j + \mathbf{U}_i \mathbf{h}_{j-1} + \mathbf{b}_i), \tag{4}$$

$$\mathbf{o}_j = \sigma(\mathbf{W}_o \hat{\mathbf{v}}_j + \mathbf{U}_o \mathbf{h}_{j-1} + \mathbf{b}_o), \tag{5}$$

$$\mathbf{C}_j = \mathbf{f}_j \odot \mathbf{C}_{j-1} + \mathbf{i}_j \odot \widetilde{\mathbf{C}}_t, \tag{6}$$

where $\odot$ denotes the element-wise product and $\widetilde{\mathbf{C}}_j$ is the candidate cell state at the j-th step defined as

$$\widetilde{\mathbf{C}}_t = \tanh(\mathbf{W}_c \hat{\mathbf{v}}_j + \mathbf{U}_c \mathbf{h}_{j-1} + b_c). \tag{7}$$

The matrices $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o \in \mathbb{R}^{M \times D}$ and $\mathbf{U}_f, \mathbf{U}_i, \mathbf{U}_o \in \mathbb{R}^{M \times M}$ in Equations (3)-(5) are transformation matrices, while the vectors $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o \in \mathbb{R}^M$ represent the corresponding bias terms. Finally, the hidden state at step j is computed as a function of the output and cell states:

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh(\mathbf{C}_j). \tag{8}$$

The procedure described through Equations (3)-(8) can be also formulated as a function in the following manner:

$$\left[\hat{\mathbf{h}}_j, \hat{\mathbf{C}}_j\right]_{j=1}^{L+1} = f_{LSTM}\left(\{\hat{\mathbf{v}}_j\}_{j=1}^{L+1},\ \left[\mathbf{h}_j, \mathbf{C}_j\right]_{j=1}^{L+1}\right), \tag{9}$$

where $\mathbf{h}_j$ is initially generated using an orthogonal initialization, while $\mathbf{C}_j$ is initialized as a mixture of uniform and orthogonal random values. Note that the hidden state vector calculated at the last step $(L+1)$ is taken to summarize the entire input sequence into a sequence (*user*) embedding $\mathbf{h} = \mathbf{h}_{L+1}$.

3.4 Sequence Reconstruction

Given a learned sequence embedding $\mathbf{h}$, the decoder component of TASA seeks to reconstruct the input sequence from $\mathbf{h}$. First, the *start* token a_{start} is appended at the beginning of the original sequence, i.e. $\{a_0 = a_{start}, a_1, \ldots, a_L\}$. Correspondingly, $\tau_0 = 0$ is taken as the stop feature for a_0, resulting in $\{\tau_0, \tau_1, \ldots, \tau_L\}$. In the decoding, each activity a_j is used to predict a_{j+1} starting from the *start* activity, i.e. for $j = 0, \ldots, L$. For this purpose, the activities, including a_0, are mapped into their corresponding embeddings $\mathbf{v}_j$ and thereafter multiplied by the temporal scores δ_j. The resulting embeddings $\hat{\mathbf{v}}_j$ are passed to an LSTM network

$$\left[\check{\mathbf{h}}_j, \check{\mathbf{C}}_j\right]_{j=0}^{L} = f_{LSTM}\left(\{\hat{\mathbf{v}}_j\}_{j=0}^{L},\ \left[\hat{\mathbf{h}}_{L+1}, \hat{\mathbf{C}}_{L+1}\right]_{j=0}^{L}\right), \tag{10}$$

in which the initial states are set to the last hidden and cell states $\left[\hat{\mathbf{h}}_{L+1}, \hat{\mathbf{C}}_{L+1}\right]$ learned by the encoder component.

The reconstruction problem is treated as a multi-class classification problem in which each activity a_{j+1} (from the original sequence) is predicted from the decoder outputs $\check{\mathbf{h}}_j$ by passing them to a fully-connected layer

$$\mathbf{o}_{FC,j+1} = \left(\check{\mathbf{h}}_j^\top \mathbf{W}_{FC} + \mathbf{b}_{FC}\right). \tag{11}$$

The outputs from the fully-connected layer are then passed through a softmax activation function to calculate the probabilities for each activity:

$$P(a_{j+1} = a^{(k)} | \check{\mathbf{h}}_j; \mathbf{W}_{FC}) = \frac{\exp(\mathbf{o}_{FC,k})}{\sum_j \exp(\mathbf{o}_{FC,j})}, \tag{12}$$

where $a^{(k)} \in V$. Note that computing Eq. (12) can be quite expensive when the number of unique activities $|V|$ is large. In this work, the exact softmax value $P(a_{j+1} = a^{(k)} | \check{\mathbf{h}}_j; \mathbf{W}_{FC})$ was calculated for all unique activities $a^{(k)} \in V$ due to the availability of computational resources. Nevertheless, in case of limited resources or a need for more efficient model training, we suggest approximating Eq. (12) by computing it over a subset of sampled activities using candidate sampling [18].

Parameter Learning. Given a set $\mathcal{S} = \{a_j^n\}_{j=1,n=1}^{L^n,N}$ of N activity sequences, a logistic loss ℓ^n is defined for the n-th sequence as

$$\ell^n = -\sum_{j=0}^{L^n} \sum_{k=1}^{|V|} I(a_{j+1}^n = a^{(k)}) \log P(a_{j+1}^n = a^{(k)} | \check{\mathbf{h}}_j; \mathbf{W}_{FC}) + I(a_{j+1}^n \neq a^{(k)}) \log(1 - P(a_{j+1}^n = a^{(k)} | \check{\mathbf{h}}_j; \mathbf{W}_{FC})), \tag{13}$$

where I is an indicator function. The parameters of TASA are then learned by minimizing the total loss $\mathcal{L} = \sum_{n=1}^{N} \ell^n$.

For a graphical illustration of TASA's building blocks (described above in Sections 3.1-3.4), refer to Figure 1.

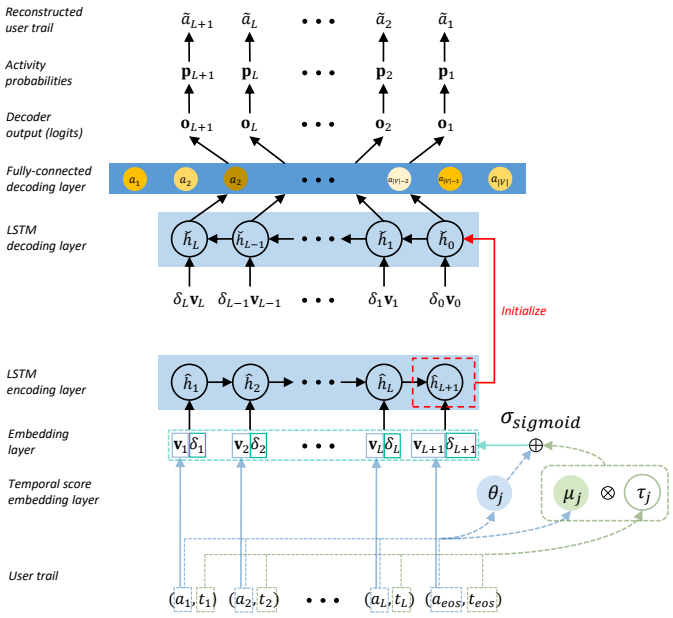

Figure 1: TASA model architecture.

4 DATA

4.1 RecSys 2015 Challenge Data

We conducted purchase prediction experiments on a publicly available dataset obtained from the RecSys Challenge in 2015. This dataset contains sequences (sessions) of click events with respective timestamps from the Yoochoose website. Some sessions ended with a purchase event (if so, the label was set as positive, otherwise negative). There are 3, 985, 870 sessions in the training and 442, 167 in the test dataset, out of which 4, 050, 782 are labeled as negative, and 377, 255 are labeled as positive. Trails were pre-processed such that the maximum sequence length was set to 100 (the last 100 events were kept, otherwise, the trails were padded). The trails containing less than 3 activities were filtered out and only the activities having more than 5 occurrences in the whole dataset were kept, which resulted in 52, 739 unique activities in the final vocabulary.

4.2 User Activity Trails from Verizon Media

We also conducted experiments using user activity trails data from Verizon Media (VM). This includes activities chronologically collected for users, derived from heterogeneous sources, e.g., Yahoo Search, commercial email receipts, reading news and other content on publishers' webpages associated with Verizon Media such as the Yahoo homepage, Yahoo Finance, Sports and News, advertising data (e.g., ad impressions and clicks), etc. A representation of an activity comprises of activity ID, timestamp, its type (e.g., search, invoice, reservation, content view, order confirmation, parcel delivery), and a raw description of the activity (e.g., the exact search query for search activities). All personally identifiable information (PII) was stripped upstream from the datasets used in this study.

Millions of activity trails were collected (i.e. user identifiers; note that a unique user id does not necessarily map to a single user). All unique activities performed by the users were sorted by frequency and the 200,000 most frequent were selected, i.e. $|V| = 200,000$. For the purposes of supervised evaluation, conversions (labels) were collected for over a one-month period for 3 different advertisers. A trail is labeled as positive if its corresponding user converted for a specific advertiser soon after the last event in the trail. For each advertiser, the user trails were divided into training, validation and test sets. Note that the user trails were sampled from a larger pool of user trails such that they correspond to the same set of users for all three advertisers. After eligible users and events were selected, negative downsampling was performed to maintain roughly 5% of the positives for all advertisers.

5 EXPERIMENTS

This section describes the baseline models, the experimental design and defines the evaluation metrics used in the experiments. Thereafter, the experimental results are presented and discussed.

5.1 Experimental Setup

5.1.1 Baselines. The baseline models considered in the conducted experiments are described as follows:

- *Fully-Connected Autoencoder* (AE): A conventional autoencoder that uses fully-connected layers for encoding and decoding.
- *Seq2seq Autoencoder* (seq2seq) [38]: A sequential autoencoder in which an LSTM encodes a set of sequences into fixed-length vector representations, followed by another LSTM aiming to reconstrcut the original sequences from the vector representations.
- *Time-Aware Seq2seq* (TA-seq2seq) [5]: A Time-Aware LSTM (T-LSTM) autoencoder that accounts for irregular time gaps between consecutive sequence elements.
- *Integrated Sequence Autoencoder* (ISA) [32]: An autoencoding framework that integrates two classical mechanisms to account for both (1) the global silhouette information that underlies a sequence and (2) the local temporal dependencies, which allows for comparing entire sequences, not only their constituents. ISA also captures irregular time gaps by introducing *stop features* to serve as temporal stamps in the sequence reconstruction process.
- ∗ *Logistic Regression* (LR): A logistic regression model that learns to predict a binary target variable by minimizing the logistic loss.
- ∗ *Attention-based RNN* (attRNN) [42]: An attention-based recurrent neural network designed specifically for conversion prediction from user activity trails.
- ∗ *eXtreme Gradient Boosting* (XGBoost) [8]: A scalable, distributed gradient tree boosting algorithm.

The summaries of the supervised baselines are marked with "∗".

5.1.2 Evaluation metrics. The considered autoencoder approaches learn in an unsupervised manner, yet they are applied to multiple supervised tasks. Hence, both unsupervised and supervised metrics were utilized to assess the performance of each approach.

Unsupervised metrics. The unsupervised metrics used in the experiments are summarized as follows:

- *Reconstruction Accuracy* measures the fraction of activities in a reconstructed (output) sequence $\{\tilde{a}_1, \ldots, \tilde{a}_{\tilde{L}}\}$ that appear on the same positions as in the original (input) sequence $\{a_1, \ldots, a_L\}$: $R = \frac{1}{L} \sum_{j=1}^{L} I(a_j = \tilde{a}_j)$.

- *ROUGE (Recall-Oriented Understudy for Gisting Evaluation)* [25] is an n-gram based measure of the quality of a candidate (output) summary with respect to one or multiple reference (input) summaries. Although initially proposed for text summarization, ROUGE is used in this work for measuring the n-gram recall between a single input sequence of activities and its corresponding reconstructed (output) sequence, i.e.
$\text{ROUGE}_n = \frac{\frac{1}{2}\sum_{j=1}^{L-n+1}\sum_{k=1}^{\tilde{L}-n+1} I(\{a_j,\ldots,a_{j+n-1}\}=\{\tilde{a}_k,\ldots,\tilde{a}_{k+n-1}\})}{L-n+1}$.
Note that all three ROUGE_1, ROUGE_2 and ROUGE_w are measured, where w denotes the Longest Common Subsequence (LCS) in the input sequence.
- *BLEU (BiLingual Evaluation Understudy)* [30]: Solely measuring recall over the n-grams in an input sequence favors long output sequences. Therefore, BLEU evaluates the quality of the input sequence reconstruction based on the n-gram precision (instead of the n-gram recall) between the input and output sequences, defined as
$\text{precision}_n = \frac{\frac{1}{2}\sum_{k=1}^{\tilde{L}-n+1}\sum_{j=1}^{L-n+1} I(\{\tilde{a}_k,\ldots,\tilde{a}_{k+n-1}\}=\{a_j,\ldots,a_{j+n-1}\})}{\tilde{L}-n+1}$.
Finally, the BLEU score is calculated by combining the average logarithms of the precision scores with exceeded length penalization (for more details, the reader is referred to [30]).

Supervised metrics. The supervised tasks were formulated as classification problems, hence the common performance indicator of area under the ROC curve (AUC) was used for evaluation based on the label probability scores estimated by the models.

Reproducibility notes. The activity and user embedding dimensions were set to 100. The activity embeddings were initialized to random uniform numbers in $[-0.05, 0.05]$, while for the approaches utilizing LSTM the sequential embeddings were initialized using an orthogonal initialization with a multiplicative factor of 1. The proposed model, along with the baselines, were trained for 10 epochs using the Adam [20] optimizer with a learning rate of 0.001. The gradient updates on the proprietary and public datasets were performed using batches of size 32 and 64, respectively. All approaches were implemented in Python 3.6 using TensorFlow 1.13.1 and run on a distributed TensorFlowOnSpark[1] infrastructure. Each approach was run on 100 Spark executors, while its parameters were distributed among 5 executors.

5.2 Results

We conducted two sets of experiments, focusing on (1) the reconstruction capability of the considered approaches, and (2) assessing the embedding quality with respect to several supervised tasks.

Sequence reconstruction capability. User embeddings were learned from the user trails available in both the proprietary and public datasets, using each of the *unsupervised* approaches summarized in Section 5.1.1. Upon training, the sequence reconstruction capability of the approaches was evaluated on a separate hold-out test set using the unsupervised measures defined in Section 5.1.2.

First, we evaluated the proposed TASA against the baselines on the sequence reconstruction task, using the proprietary VM activity data and reported the results in Table 2. Table 2 suggests that TASA achieves greater performance compared to the other approaches,

[1]https://github.com/yahoo/TensorflowOnSpark, accessed June 2020.

Table 2: Sequence reconstruction performance on the VM proprietary dataset.

Model	Rec. Acc.	BLEU_n $n=1$	BLEU_n $n=2$	ROUGE_n $n=1$	ROUGE_n $n=2$	ROUGE_n $n=w$
AE	0.0451	0.0451	0.0000	0.0115	0.0000	0.0451
seq2seq	0.3407	0.4734	0.2209	0.4389	0.1825	0.4192
TA-seq2seq	0.3731	0.5018	0.2524	0.4725	0.2159	0.4478
ISA	0.2839	0.4102	0.1674	0.3796	0.1336	0.3620
TASA	0.3761	0.5103	0.2538	0.4846	0.2169	0.4511

Table 3: Sequence reconstruction performance on the (public) RecSys 2015 challenge dataset.

Model	Rec. Acc.	BLEU_n $n=1$	BLEU_n $n=2$	ROUGE_n $n=1$	ROUGE_n $n=2$	ROUGE_n $n=w$
AE	0.0136	0.0136	0.0085	0.0122	0.0082	0.0136
seq2seq	0.1235	0.2396	0.0709	0.2551	0.0742	0.2254
TA-seq2seq	0.1725	0.2664	0.0936	0.2807	0.0965	0.2527
ISA	0.1979	0.2535	0.0927	0.2851	0.0991	0.2464
TASA	0.5244	0.5500	0.3952	0.5691	0.4012	0.5441

across all sequence reconstruction measures. Further, it shows that approaches which take into consideration the temporal aspect of the data, such as TA-seq2seq and TASA, do better at capturing the information needed for more accurate sequence reconstruction than the rest of the approaches. Nonetheless, seq2seq greatly outperforms AE, indicating that the sequential information modeling performed by LSTM autoencoders seems to be an important basis for capturing the sequential nature of the user trails.

Reconstruction evaluation was also performed on the public dataset. Table 3 summarizes the results. The performances of AE, seq2seq, TA-seq2seq and TASA on this dataset seem to be somewhat consistent with their performances on the proprietary dataset. Once again, seq2seq outperforms AE and the time-aware variants manifest improvements over the ones that do not preserve time. ISA specifically exhibits better performance relative to the other approaches, being the second-best performing model. TASA, however, consistently outperforms the other alternatives across all evaluation measures, here showing even larger performance improvements than those obtained on the proprietary data.

Supervised predictive tasks. The learned vector representations were further tested on supervised predictive tasks. For the proprietary datasets, a baseline approach was used to build predictive models using users' demographic data and the lift in AUC (expressed in percentages) was computed for the rest of the models with respect to this baseline. The user embeddings learned from the unsupervised models were appended to these demographic data and an LR model was learned on the resulting concatenated features. In addition, two more supervised approaches were compared: an LR learned on 1-hot encoded activity features and an attRNN learned on user sequences, both concatenated with user demographics. The lifts in AUC obtained on the proprietary datasets are given in Table 4. It should be noted that the lifts are expressed in percentages

Table 4: All LR models from the left section of the table are trained on (1) the user demographic features concatenated with (2) the user embeddings generated by the corresponding unsupervised embedding model given in the parentheses of LR(·). The AUC lifts in the left section are calculated relatively to an LR model trained solely on the user demographic features. As for the right section of the table, LR 1-hot is an LR model trained on one-hot encoded features (extracted from the user trails), while attRNN is trained on the original user trails, concatenated to the TASA-generated user embeddings in both cases. The corresponding lifts in AUC introduced by the former and the latter are calculated relatively to an LR model trained solely on the one-hot encoded features and an attRNN trained solely on the original user trails, respectively.

Supervised Task	LR(AE)	LR(seq2seq)	LR(TA-seq2seq)	LR(ISA)	LR(TASA)	LR 1-hot	attRNN
Advertiser A	27.13%	29.68%	29.73%	29.96%	30.44%	22.78%	30.46%
Advertiser B	11.79%	15.99%	15.91%	18.24%	20.34%	10.65%	20.18%
Advertiser C	-18.88%	18.63%	16.07%	16.44%	20.37%	7.66%	20.49%

Table 5: Predictive performance in terms of AUC on the (public) RecSys 2015 challenge dataset. For more details on the models presented in the left and right sections of the table, refer to the caption of Table 4.

Supervised Task	LR(AE)	LR(seq2seq)	LR(TA-seq2seq)	LR(ISA)	LR(TASA)	LR 1-hot	attRNN
RecSys 2015 Challenge	0.5555	0.6022	0.7523	0.7420	0.7563	0.7277	0.7591

since any disclosure of the AUC values is not allowed according to internal corporate policies. Nevertheless, despite the challenging real-world nature of the predictive tasks on the proprietary data, it was observed that the AUC values obtained by all models are legitimate in the sense that they are substantially greater than 0.5.

It is clear that using only the demographic data is not sufficient. LR trained on the one-hot encoded user activities improves this performance. Nevertheless, once the user embeddings are concatenated to the demographic features, the improvements in AUC are evident. In particular, the proposed TASA achieves an improvement of $> 25\%$ in AUC on Advertiser A, $> 10\%$ on Advertiser B, and $> 15\%$ on Advertiser C. This indicates that, although LR is trained in a supervised manner, when trained on unsupervised user embeddings it still attains greater predictive performance. This is due to the sequential information within the user trails that is being captured by the sequential autoencoder variants, but not by LR trained on a simple one-hot transformation of the user activities.

However, LR achieves its highest performance when trained on TASA's user embeddings, rather than on the embeddings produced by any other autoencoder variant. It even achieves comparable performance to the supervised attRNN model. This is of considerable importance since TASA learned the user embeddings not being aware of the prediction target as opposed to attRNN that was tailored for each task separately. In addition, attRNN requires to be retrained for each task which consumes a significant amount of time. Therefore, TASA has a great advantage since it can be trained offline once and readily applied across multiple prediction tasks, allowing for subsequently leveraging low-latency models (to accommodate serving time) which, if needed, can be efficiently retrained on TASA's low-dimensional embeddings.

The approaches were evaluated on the public RecSys 2015 dataset as well. Consistent with the previous observations, LR(TASA) convincingly outperformed the competitive autoencoder approaches, while being almost as good as attRNN. In fact, all models that account for sequential and temporal data aspects outperformed the LR variant trained on the one-hot encoded sequences (see Table 5). TASA learned embeddings that again resulted in the model with the highest AUC among the LR models trained on user embeddings. This indicates that TASA's user embeddings appear to be more informative than the ones generated by the alternative autoencoders.

6 SERVICE PIPELINE

In Section 5, it has been observed that the proposed TASA outperforms the other unsupervised embedding baselines on both the sequence reconstruction task and the subsequent supervised tasks. Therefore, TASA is used within a service system designed to provide *time-aware (user) embeddings as a service*. Generally speaking, the service takes a collection of sequences as an input and periodically generates time-aware embeddings of the sequences and their constituent elements. Although the service is clearly applicable to any type of sequence data, in this work it is described from the perspective of *embedding sequences of user activities*, i.e. *user trails*. The stages of the service system pipeline are described as follows:

- **Stage 1: Generation of Sequences (User Trails)**. The activities that users performed are first queried from multiple heterogeneous data sources. In our case, these include Yahoo Search queries, commercial email receipts, reading news and other content on publishers' webpages associated with Verizon Media such as the Yahoo homepage, Yahoo Finance, Sports and News, advertising data (e.g., ad impressions and clicks), etc. After stripping all personally identifiable information (PII) from the raw description of each activity (e.g., the exact search query for search activities), the frequency of each unique activity is calculated. An index is then assigned to each activity among the top-K most frequent activities (we set $K = 200,000$) and the mappings between the activities' descriptions and their indices are organized in a vocabulary V.
- **Stage 2: Model Training.** The proposed TASA model takes (1) the extracted user trails, (2) the timestamps of their constituent activities and (3) the activity vocabulary V as an input, and learns

an M-dimensional time-preserving embedding for each user trail following the procedure described in Section 3. The resulting M-dimensional (we set $M = 100$) continuous vectors are stored in a centralized database. Similarly, the learned parameters of TASA are also stored in a separate database for further use.

- **Stage 3: Embedding Incoming Activities/Users.** Upon model training, the learned TASA parameters can be fetched and used to embed incoming activities or entire user trails. Note that if an incoming activity turns out to be one of the trending activities, meaning that it was performed by a large number of users and passes the frequency threshold, it will be included in V and considered for the next scheduled model training. However, the incoming activity may not be present in the vocabulary V. In this case, the new activity is not immediately added to V, but it is included in V the next time V is being updated. Similarly, incoming users are embedded by first embedding the activities from their trails that are present in V, while considering the currently unknown activities for inclusion in the next vocabulary update cycle. Consequently, the vocabulary is updated periodically (e.g., once a week), while the trails are typically updated more frequently (on a daily basis, for instance).

It should be stressed that the service is general enough to be utilized for different types of downstream tasks in the realms of advertising, user profiling, or recommendation. These tasks may include click/conversion prediction, user segmentation, lookalike modeling (e.g., retrieving the top-10 most similar users to a given user based on embedding similarity; or leveraging the user embeddings to support other autoencoder approaches for lookalike modeling [12]), recommendation of news, products or ads, among others. This allows for individual teams to carry out their downstream tasks without being burdened with efforts around understanding the data, crafting features, etc.

A bird's-eye view of the entire service pipeline, encompassing the aforedescribed stages, is presented in Figure 2.

6.1 Pipeline Evaluation

The *Embeddings-as-a-Service* pipeline is currently deployed as an internal tool which several teams utilize as a source of additional user features, mainly for offline experimentation. In Table 6 we present the results from the offline experiments for four sets of audiences on a conversion prediction task. The results are expressed in terms of percentage AUC lifts as the disclosure of the AUC values is not allowed according to internal corporate policies. However, we have verified that, despite the challenging nature of predicting conversions for multiple real-world audiences, the original AUC values are legitimate in the sense that they are substantially greater than 0.5. The relative improvements in AUC were obtained by comparing a supervised model that utilizes the TASA-generated embeddings in addition to the current production features, to a model built using only the current production features. In both cases, XGBoost (a gradient boosting algorithm summarized in Section 5.1.1) is used for supervised learning. Therefore, the percentage improvements for all four audiences are a result of the TASA-generated user embeddings being utilized along with the production features. It should be stressed that the production system is a very strong baseline as it utilizes a set of features that have been refined over many years and are already achieving high predictive performance. Considering that the performance evaluation is carried out on audiences that consist of over a billion users each, even a lift of about 0.5% is substantial and can make a difference given the high volume of predictions the model is expected to make.

Table 6: AUC lifts introduced by the *Embedding-as-a-Service* pipeline on four in-house conversion prediction tasks.

Task	AUC Lift
Audience 1	0.92%
Audience 2	0.09%
Audience 3	0.95%
Audience 4	0.40%

7 CONCLUSION

Motivated by the goal of reducing the redundancy in feature engineering effort for multiple user-centered tasks, we explored the capabilities of contemporary unsupervised sequence embedding methods. Moreover, we proposed TASA, a sequential autoencoder that extends the modeling capacity of conventional sequence embedding approaches to allow for modeling the influence of temporal information within user sequences. The experimental evaluation on both proprietary and public data demonstrated that TASA outperforms alternative autoencoder variants when it comes to sequence reconstruction, and leads to improved predictive performance when used in supervised tasks. Further, we presented a service pipeline that leverages TASA to generate time-aware user embeddings. When used along with the current production features, the pipeline-generated embeddings resulted in improved conversion prediction performance on four different audiences. Finally, we expect the pipeline to serve as a powerful tool for supporting various tasks vital for digital media company businesses.

ACKNOWLEDGMENTS

Martin Pavlovski and Zoran Obradovic's research was supported in part by the NSF grant IIS-8142183 and by Yahoo! Research "Faculty Research and Engagement Program". We would also like to thank the engineering teams within Verizon Media that helped develop the data and evaluation pipelines.

REFERENCES

[1] Shahin Amiriparian, Michael Freitag, Nicholas Cummins, and Björn Schuller. 2017. Sequence to sequence autoencoders for unsupervised representation learning from audio. In *Proc. of the DCASE 2017 Workshop*.

[2] Sarath Chandar AP, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*. 1853–1861.

[3] Tian Bai, Shanshan Zhang, Brian L Egleston, and Slobodan Vucetic. 2018. Interpretable representation learning for healthcare via capturing disease progression through time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 43–51.

[4] Dana H Ballard. 1987. Modular Learning in Neural Networks.. In *AAAI*. 279–284.

[5] Inci M Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K Jain, and Jiayu Zhou. 2017. Patient subtyping via time-aware LSTM networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 65–74.

[6] Sean Billings. 2018. Gradient Augmented Information Retrieval with Autoencoders and Semantic Hashing. *arXiv preprint arXiv:1803.04494* (2018).

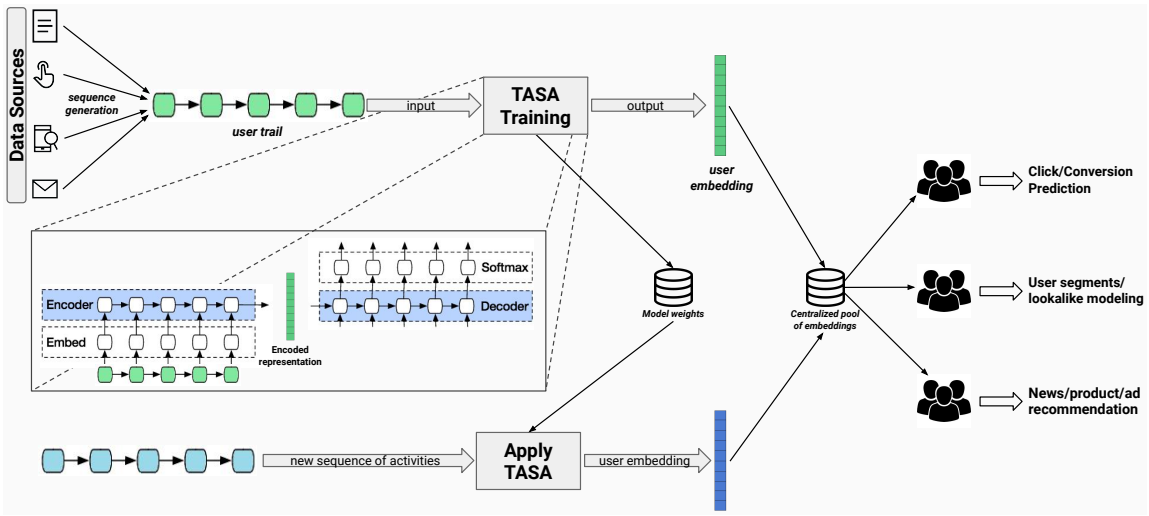

Figure 2: A bird's-eye view of the *Embeddings-as-a-Service* pipeline.

[7] Hervé Bourlard and Yves Kamp. 1988. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics* 59, 4-5 (1988), 291–294.

[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 785–794.

[9] Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. *arXiv preprint arXiv:1603.00982* (2016).

[10] Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*. 3079–3087.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[12] Khoa D Doan, Pranjul Yadav, and Chandan K Reddy. 2019. Adversarial Factorization Autoencoder for Look-alike Modeling. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2803–2812.

[13] Djordje Gligorijevic, Jelena Gligorijevic, and Aaron Flores. 2019. Time-Aware Prospective Modeling of Users for Online Display Advertising. *AdKDD 2019 workshop at the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019).

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[15] Zhicheng He, Jie Liu, Na Li, and Yalou Huang. 2019. Learning Network-to-Network Model for Content-rich Network Embedding. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1037–1045.

[16] Geoffrey E Hinton and Sam T Roweis. 2003. Stochastic neighbor embedding. In *Advances in neural information processing systems*. 857–864.

[17] Geoffrey E Hinton and Richard S Zemel. 1994. Autoencoders, minimum description length and Helmholtz free energy. In *Advances in neural information processing systems*. 3–10.

[18] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).

[19] Ian Jolliffe. 2011. *Principal component analysis*. Springer.

[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[21] Alex Krizhevsky and Geoffrey E Hinton. 2011. Using very deep autoencoders for content-based image retrieval.. In *ESANN*, Vol. 1. 2.

[22] Yann Le Cun. 1986. Modèles connexionnistes de l'apprentissage. *These de Doctorat. Universite Paris* (1986).

[23] Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (1999), 788.

[24] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057* (2015).

[25] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[26] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. 2014. Autoencoder for words. *Neurocomputing* 139 (2014), 84–96.

[27] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114* (2015).

[28] Geoffrey McLachlan. 2004. *Discriminant analysis and statistical pattern recognition*. Vol. 544. John Wiley & Sons.

[29] Al Mead. 1992. Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society: Series D (The Statistician)* 41, 1 (1992), 27–39.

[30] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[31] Viorica Patraucean, Ankur Handa, and Roberto Cipolla. 2015. Spatio-temporal video autoencoder with differentiable memory. *arXiv preprint arXiv:1511.06309* (2015).

[32] Wenjie Pei and David MJ Tax. 2018. Unsupervised Learning of Sequence Representations by Autoencoders. *arXiv preprint arXiv:1804.00946* (2018).

[33] Jonas Pfeiffer, Samuel Broscheit, Rainer Gemulla, and Mathias Göschl. 2018. A neural autoencoder approach for document ranking and query refinement in pharmacogenomic information retrieval. Association for Computational Linguistics.

[34] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. 2016. Deepcare: A deep dynamic memory model for predictive medicine. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 30–41.

[35] Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science* 290, 5500 (2000), 2323–2326.

[36] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.

[37] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised learning of video representations using lstms. In *International conference on machine learning*. 843–852.

[38] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[39] Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290, 5500 (2000), 2319–2323.

[40] Vincent Wan, Yannis Agiomyrgiannakis, Hanna Silen, and Jakub Vit. 2017. Google's Next-Generation Real-Time Unit-Selection Synthesizer Using Sequence-to-Sequence LSTM-Based Autoencoders.. In *INTERSPEECH*. 1143–1147.

[41] Yuan Zhang, Xi Yang, Julie Ivy, and Min Chi. 2019. ATTAIN: Attention-based Time-Aware LSTM Networks for Disease Progression Modeling. IJCAI.

[42] Yichao Zhou, Shaunak Mishra, Jelena Gligorijevic, Tarun Bhatia, and Narayan Bhamidipati. 2019. Understanding consumer journey using attention based recurrent neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 3102–3111.