ClickGraph: Web Page Embedding using Clickstream Data for Multitask Learning

Porter Jenkins
Pennsylvania State University
prj3@psu.edu

ABSTRACT

The rise of big data frameworks has given website administrators the ability to track user clickstream data with more detail than ever before. These clickstreams can represent the user's intent and purpose in visiting the site. While existing work has explored methods for predicting future user actions, these methods are limited focus solely on one task at a time, ignore graph structure inherent in clickstreams, or model the conversion of the entire clickstream session, ignoring complexities such as multiple conversions in a single session. In this work, we formulate the novel problem of simultaneously predicting future user actions given a user's clickstream history. We argue that clickstream data contains important signal for predicting future user action. To tackle this new problem, we propose a novel method called ClickGraph, a recurrent neural network that encodes the graph structure of user click trajectories in the learned representations of web pages. We conduct experiments on a real-world dataset and demonstrate that this multitask learning approach is effective at improving the prediction of form fill conversions over strong baselines. In particular, we demonstrate that the ClickGraph model is effective at reducing false positive rates, increasing F1 scores, and improving recall.

KEYWORDS

Clickstreams, Recurrent neural networks, Representation learning, Multitask learning

ACM Reference format:
Porter Jenkins. 2019. ClickGraph: Web Page Embedding using Clickstream Data for Multitask Learning. In *Proceedings of Companion Proceedings of the 2019 World Wide Web Conference, San Francisco, CA, USA, May 13–17, 2019 (WWW '19 Companion),* 5 pages.
https://doi.org/10.1145/3308560.3314198

1 INTRODUCTION

Website administrators are keenly interested in understanding the behavior of visitors, or users, of their site. Better understanding of website traffic typically leads to better site design, increased sales, or improvement of other business goals. A litany of research in recommendation systems [17][13][2] and click-through-rate prediction [8] [16] relies on understanding or statistically leveraging past user behavior. The goal of many of these algorithms is to predict future user actions. If a program can successfully predict future user action, it would then be able to make intervention, and presumably drive improvement in some business outcome.

WWW '19 Companion, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6675-5/19/05.
https://doi.org/10.1145/3308560.3314198

One important data feature to consider when predicting user actions is the clickstream. In this paper, we refer to a clickstream as a sequence of web pages a user visits. These pages are connected by a set of links, in a network- or graph-like structure. In many contexts, this clickstream data captures important characteristics about the user, and may often contain signal relevant to predicting future action. For example, [3] hypothesizes visitors to an e-commerce site augment their knowledge about the offered products as they traverse the site. We adopt this view, and believe it can be used to predict a variety of future user actions.

We extend this avenue of research and study the relationship between clickstream behavior and form-fill conversion rate. A form-fill is an event where the user provides information for future marketing contact. Many companies maintain a page where traffic from paid or organic search is directed. This page is designed to increase interaction with potential customers and generate new leads [5].

To the best of our knowledge, no existing work seeks to jointly model multiple user actions in clickstream data. In this work, we propose the novel problem of simultaneously predicting multiple user actions (multi-task learning) given sequential browsing history (clickstreams). Simultaneous prediction of future action allows for a multi-dimensional understanding of future behavior, and better informs understanding of user intent and preferences. There are significant challenges associated with solving this problem.

(1) It is difficult to effectively model the clickstream sequence and the relationships between different actions. For one, representing sequences of web pages in a meaningful way is a challenge. Many data scientists and researchers rely on naive representations of clickstream data such as discrete counts of site visits, number of pages clicked, session duration, average page duration, etc... [3]. Additionally, we cannot simply treat the different actions as independent. One action likely informs another.
(2) If independence is not a valid assumption, what is the best way to jointly model user actions, and in turn make simultaneous predictions? This is a non-trivial task that has yet to be solved in the literature.

We solve the problems in (1) and (2) by proposing a new method, `ClickGraph`, that is an LSTM-based model. As a Recurrent Neural Network (RNN), `ClickGraph` naturally models the variable-length, sequential nature of clickstream data. The model simultaneously predicts the next page the user will visit, the time duration, and form fill conversion. In this multi-task learning approach, we model the joint probability of actions by factoring conditional likelihoods in a way that mirrors the user's decision process. Additionally, we incorporate graph structure into the loss of the next page prediction

task, which informs other tasks of interest, such as form fill conversion. Finally, we use an embedding layer to learn representations of web pages to avoid manual, and often ineffective extraction of features.

We validate our proposed method against a variety of strong baselines and demonstrate that it is more effective at predicting form fill conversion. Conversion rates in a targeted marketing setting are notoriously low, where the average global conversion rate is approximately 2-3%, and the average cost (per dollar spent) to acquire a new customer is 92 cents [3]. Therefore, even marginal improvements in accuracy, or decreases in false negatives would have a dramatic impact on revenue. We show that, in addition to improving accuracy and AUC, `ClickGraph` is highly effective at decreasing false negative (FN) classifications and improving recall, two key metrics used in assessing the prediction of rare events.

In summary, the main contributions of our paper are as follows:

- We define a new problem: simultaneously predicting future user actions from sequential, clickstream data. Simultaneous prediction improves single tasks of high interest, such as form fill conversion prediction
- We solve this problem by proposing a new method, `ClickGraph`, that incorporates the graph structure of websites via a multi-task loss function. The model learns representations of web pages that encode this network information, and shares statistical strength across multiple user actions. These representations can be used by a machine learning model to better understand user intent.

2 RELATED WORK

Our problem touches on two separate literature streams, that both fall under of the umbrella of digital marketing: click-through rate prediction and clickstream models.

2.1 Click-through Rate Prediction Models

We discuss two related areas: sponsored search and content advertising [4]. In sponsored search, the user enters a query into a search engine and is served both organic search results, and relevant ads. In content advertising, ads are displayed alongside original content of a website, where the ads can either be generic or targeted to the user [4]. Rather than predict ad click-through rates, we address the problem of predicting future actions a user will take within a site.

Recent work focuses on learning representations of users using neural networks in an Embedding&MLP approach [8] [13] [8] for predicting digital ads clicks. Additionally, some approaches model sequential data [16]

2.2 Clickstream Models

Historically, there are four common approaches to model clickstream data of web site visitors: sequential pattern analysis, association rules, clustering, and Markov models [7]. All of these methods model a conversion event (click, buy, form-fill, etc.) for an entire sequence of clickstream data for any given session. The granular temporal information of conversion, or even decoding multiple conversions in a single session, is unaccounted for in these methods.

To the best of our knowledge, no work to date tries to solve the problem of simultaneously predicting multiple user actions from sequential clickstream data.

3 PROBLEM DEFINITION

In what follows, we formally define our problem in terms of inputs and outputs. The input to our problem consists of three major components, a website graph, user clickstreams, and user actions.

3.1 Inputs

Definition 3.1 (Website graph). We treat the website of interest as a graph, $\mathcal{G}$, with nodes, or webpages, $\mathcal{P} = \{p_1, p_2, ..., p_m\}$ and edges, $\mathcal{E} = \{e_1, e_2, ..., e_q\}$. A given web page, p_i, is associated with a set of adjacent pages, or adjacency list, $\mathcal{A}_i = \{p_1, p_2, ..., p_k\}$.

Definition 3.2 (User actions). Additionally, we observe a set of n users, $\mathcal{U} = \{u_1, u_2, ..., u_n\}$, and user actions $\mathcal{Y} = \{Y_{1,1}, ..., Y_{n,t}\}$. For each of the n users, at each timestamp, t, we observe a tuple of actions $Y_{it} = \langle y_{it}, d_{it}, p_{i,t+1} \rangle$, where:

- y_{it}: (*Form fill*) Indicator denoting whether the user requests marketing information (fills out a form). Where, $y_{it} \in \{0, 1\}$
- d_{it}: (*Duration*) Time (in seconds) user u_i spends on page p_m. Where $d_{it} \in \mathbb{R}^+$
- $p_{i,t+1}$: (*Next page*) The future page user u_i will visit at time $t + 1$. Where $p_{i,t+1} \in \mathcal{A}_i$

Definition 3.3 (User clickstream). Finally, we observe a set of n clickstreams, $\mathcal{S} = \{S_1, S_2, ..., S_n\}$, where S_i is a sequence of web pages and actions at time t,

3.2 Outputs

We propose a model that takes input as described and produces probability estimates of multiple tasks. Our primary goal is to predict future user actions. Thus, the first output is a tuple of probabilities over future user actions, $\hat{\mathcal{Y}} = \{\hat{Y}_{1,t+1}, ..., \hat{Y}_{n,t+f}\}$, where $\hat{Y}_{n,t+f} = \langle P(y_{i,t+f}), P(d_{i,t+f}), P(p_{i,t+f+1}) \rangle$, and f denotes the number of future time periods predicted. As will be shown in section 4, we do not model these actions independently, but do so jointly mirroring the user's decision process.

4 PROPOSED METHOD

In this section, we provide an intuitive overview of our model, and give a very brief outline of Recurrent Neural Networks (RNN's) and Long Short-term Memory (LSTM) cells. We then provide a more in-depth discussion of our proposed method, `ClickGraph`, it's architecture, and define its loss function.

4.1 ClickGraph Overview

The `ClickGraph` model is a recurrent neural network that takes sequences of web pages as input and outputs probabilities of the three potential actions described in section 3. At each time stamp, to compute these probabilities we model the joint likelihood of all three possible actions, $P(y_{it}, d_{it}, p_{i,t+1})$. We factor this joint probability to reflect the decision process of the user. Specifically, the next page the user will visit is dependent on how long they spend on the page, and whether or not they fill a form. Additionally,

page duration is also conditional on form fill conversion. Finally, form fill conversion is dependent on the hidden state learned by the neural network. We provide a more rigorous definition in 4.3.4.

4.2 Recurrent Neural Networks and LSTM

Recurrent Neural networks are powerful deep learning models that process sequential data. Modern networks have proven effective in many tasks because they share weights across different parts of the model, and across time stamps [6]. This sharing enables better generalization and makes them more robust to examples appearing at different time stamps.

In practice, some of the most effective RNN's are those that utilize Long Short-Term Memory cells [10]. The key idea behind LSTM models is their ability to learn when to forget older states, which in turn help them alleviate the effect of gradient vanishing. LSTM's are also able to effectively manage the trade-off between long- and short-term dependencies [6].

4.3 ClickGraph Architecture

We now define the architecture of our proposed model, `ClickGraph`.

4.3.1 Input Layer. A single input to our model is a given user's clickstream, $S_i = ((p_{i,1}, Y_{i,1}), ..., (p_{i,t}, Y_{i,t}))$. At each time, t, we observe the page the user traverses to, p_i, and the set of actions taken, $Y_{i,t}$. Our goal is to predict these actions, $Y_{i,t}$, from p_i on unseen data. At each time, t, we input a one-hot representation of $\mathbf{p}_{i,t} \in \mathbb{R}^{|\mathcal{P}|}$. Note that the length of S_i varies with respect to users.

4.3.2 Webpage Embedding Layer. The input $\mathbf{p}_{i,t}$ is extremely sparse and is a poor representation of relationships between pages on the website. Pages represented as one-hot vectors all have the same euclidean distance ($\sqrt{2}$) from each other in $\mathbb{R}^{|\mathcal{P}|}$ [6]. Therefore we seek to learn a representation that encodes similarity between pages drawing on ideas from [9][16], by learning a lower dimensional page representation, $\mathbf{e}_{i,t} \in \mathbb{R}^{\delta}$, via a linear mapping:

$$\mathbf{e}_{i,t} = \mathbf{W}_{embed}\mathbf{p}_{i,t} \tag{1}$$

where $\mathbf{W}_{embed} \in \mathbb{R}^{\delta \times |\mathcal{P}|}$ is a matrix of learnable parameters.

In section 4.3.5 we define our loss function, which contains a term for next page prediction. This term incorporates the adjacency matrix of the graph in the softmax denominator. Because the weights, $\mathbf{W}_{embed}$, are updated through propagation of errors, optimization steps that minimize the loss function in turn encode graph information into the embedding vectors, $\mathbf{e}_{i,t}$.

4.3.3 LSTM Layer. The web page embedding, $\mathbf{e}_{i,t}$, is fed into an LSTM cell, which produces a hidden state at time, t. For brevity, we refer the reader to [10][6][11] and omit the LSTM definition here.

4.3.4 Multi-task Prediction Layer. Once we have obtained the hidden state vector $\mathbf{h}_t$ from the LSTM cell, we feed it into the output layer for predicting $y_{i,t}$, d_{it}, and $p_{i,t+1}$. We model the joint likelihood of the three user actions using the chain rule of probability:

$$\begin{aligned} &P(y_{i,t}, d_{it}, p_{i,t+1}) = \\ &P(y_{i,t}|h_t)P(d_{it}|y_{i,t}, h_t)P(p_{i,t+1}|d_{it}, y_{i,t}, h_t) \end{aligned} \tag{2}$$

We then encode these conditional probabilities into the architecture of the network.

Task 1: Form Fill We obtain the predicted form fill probability from equation 3

$$P(y_{i,t}|h_t) = \sigma(\mathbf{W}_{form}\mathbf{h}_t + \mathbf{b}_{form}) \tag{3}$$

Where $\sigma(\cdot)$ is the sigmoid function [6]. We then condition on this estimate of $\hat{y}_{i,t}$, to predict $P(d_{it})$ and $P(p_{i,t+1})$ as in (2).

Task 2: Page Duration

To better predict our other tasks, we first embed the predicted value for form fill into a distributed representation, $\boldsymbol{\gamma}_{it} \in \mathbb{R}^{p\times 1}$. This distributed representation eliminates the presence of a binary component of the feature vector when predicting $d_{i,t}$ and $p_{i,t+1}$. We let the dimensions of $\boldsymbol{\gamma}_{it}$ be $p = \left\lfloor \frac{\delta}{2} \right\rfloor$

$$\boldsymbol{\gamma}_{it} = \Omega\hat{\boldsymbol{y}}_{it} \tag{4}$$

where $\Omega \in \mathbb{R}^{p\times 2}$ is a matrix of learnable parameters, and $\hat{\boldsymbol{y}}_{it} \in \mathbb{R}^{2\times 1}$ is the vector of predicted probabilities for form fill conversion.

This embedded vector is concatenated with the hidden state variable, $\mathbf{h}_t$, to produce a feature vector for the regression of predicted page duration:

$$\boldsymbol{\beta}_t = \begin{bmatrix} \boldsymbol{\gamma}_t \\ \boldsymbol{h}_t \end{bmatrix} \tag{5}$$

We assume a gaussian distribution with mean, μ and unit variance for $P(d_{it})$,

$$P(d_{it}|y_{i,t}, \mathbf{h}_t) \sim \mathcal{N}(\mu, 1) \tag{6}$$

where, $\mu = \mathbf{w}_{duration}^{\top}\boldsymbol{\beta}_t + \mathbf{b}_{duration}$

We derive our predicted values for $\hat{d}_{i,t}$ from the dot product of the weight and feature vectors for page duration:

$$\hat{d}_{i,t} = \mathbf{w}_{duration}^{\top}\boldsymbol{\beta}_t + \mathbf{b}_{duration} \tag{7}$$

Task 3: Next Web Page

Finally, we concatenate the normalized predicted value, $\hat{d}_{i,t}$ (scalar), with the embedded predicted values for $\hat{y}_{i,t}$ to obtain the feature vector, $\boldsymbol{\pi}_t$, for the next page prediction task.

$$\boldsymbol{\pi}_t = \begin{bmatrix} \boldsymbol{\gamma}_t \\ \hat{d}_{i,t} \end{bmatrix} \tag{8}$$

We take the softmax of the feature vector and weight matrix over the adjacency matrix with the softmax adjacency function:

$$\hat{p}_{i,t+1} = softmax_{adj}(\mathbf{W}_{page}\boldsymbol{\pi}_t + \mathbf{b}_{page}) \tag{9}$$

Where,

$$softmax_{adj}(\mathbf{x}_i) = \frac{e^{x_i}}{\sum_{k\in\mathcal{A}_i} e^{x_k}} \tag{10}$$

and $\mathcal{A}_i$ is the adjacency list for page, p_i. This allows us to encode graphical structure into our sequential predictive model.

4.3.5 Loss Function. In the following section we define the loss function used to train ClickGraph. Our objective is to minimize the negative log likelihood in equation 11. Additionally, we insert the tuning parameter $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3]$ to control the contribution of each task to the overall loss. Using cross-entropy loss for $P(y_{i,t})$ and $P(p_{i,t+1})$, and mean squared error loss for $P(d_{it})$. We add an an L2 penalty over the weights for $p_{i,t+1}$, which is controlled by another hyperparameter, α. This additional term acts as a regularizer on the next page prediction task and is very effective at improving generalization performance in our empirical tests.

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{t=1}^{T}[\ \lambda_1 \left(y_{it}\log P(y_{i,t}) + (1-y_{it})\log(1-P(y_{i,t}))\right) + \frac{\lambda_2}{2}(\hat{d}_{it} - d_{i,t})^2 + \lambda_3 \log P(p_{i,t+1})\ ] + \frac{\alpha}{2}\|\mathbf{W}_{page}\|_2^2 \tag{11}$$

5 EXPERIMENTS

In the following section we describe the clickstream data set and experimental settings used [14] [1]. Additionally, we empirically evaluate our model against a variety of baselines, comparing against multiple metrics of performance. Finally, we provide a qualitative analysis of the embedding vectors learned by ClickGraph.

5.1 Experimental Settings

5.1.1 Dataset Description. We collect data from a large software-as-a-service company whose primary product is a data analytics platform. A portion of their site is targeted at prospective customers and is comprised of content describing the function and use of their software. We observe incoming users to this portion of the site, their clickstream trajectory, how long they spend on each page, and whether or not they click a form to provide contact information for marketing purposes. For our experiments, we structure the data in two ways. First, we maintain a tabular array of the dataset, where each transaction (row) is treated as independent from all other transactions. We rely on this data structure to feed data into baselines that ignore sequential information, or treat each observed transaction as *iid*. Additionally, we maintain a sequential data structure, where each sequence is indexed by a visit ID. The sequential data is used to easily collect sequence-based mini batches for the sequence models used in our experiments.

5.1.2 Prediction Task. We frame our prediction problem in the context of digital marketing, as discussed above. While our method can be generalized to learn about any future action facing a website user, we evaluate our model on the basis of form fill prediction, which is a binary prediction task. Form fill prediction is important because it provides insight into the user's intent.

We split our data into three sets: training (80%), validation (10%), and test (10%). For models that treat the data as *iid*, we split our tabular data structure along the rows of the design matrix. For models where sequential information is meaningful, we split our sequence data by randomly allocating *visit id*'s to the train, validation, or test set.

5.2 Quantitative Evaluation

5.2.1 Baselines. To assess the effectiveness of our model we compare to state-of-the-art baselines that vary both the sequential and graphical aspects of the data. This allows us to validate our hypothesis that incorporating graph structure into a sequential model to simultaneously predict multiple variables is effective for predicting user action. Specifically, we compare the following methods:

- **MajorityClass** A naive baseline that unconditionally predicts the majority class (e.g., $\hat{y} = 0$), regardless of sequence.
- **XGB (One Hot)** A simple classifier with no sequence or graph information. We use an XGBoost algorithm with a naive, one-hot encoding of clickstreams. Each webpage is a one-hot vector of dimension $|\ \mathcal{P}\ |$. We treat each sample of our feature matrix as *iid*.
- **XGB (Graph)** Classifier with graph information, but no sequential information. We train an XGBoost algorithm with embedded vectors of clickstreams. We rely on the DeepWalk algorithm [15] to obtain graph-based embedding vectors of webpages. We treat each sample of our feature matrix as *iid*.
- **LSTM (No Graph)** A deep learning classifer that incorporates sequential information, but graphical information. This model is a simpler form of ClickGraph, but only predicts $y_{i,t}$, and does not incorporate the graph structure of the website.
- **ClickGraph** The proposed classifier that includes both sequential and graphical information.

5.2.2 Evaluation Metrics. Given that the prediction task is binary, we utilize standard binary classification metrics.

In our data, observing a form fill event is rare; approximately 2.3% of all samples have positives labels. In the context of our problem, a key interest is in identifying true form fill events. Because a trade-off exists between false positives and false negatives [12], we weight the cost of a FN as larger than that of a FP. As such, we primarily are interested metrics such as AUC, recall, and F1. A desirable model would have the property of successfully lowering FN's, without sacrificing overall effectiveness.

5.2.3 Results. We report the results form our quantitative evaluation in table 1. Overall, we have the following three observations.

Modeling the clickstream sequence improves performance The results in table 1 demonstrate that, in general, discarding the sequential structure of the data is harmful to model performance. Clearly, the sequence models based on recurrent neural networks (LSTM (no graph) and ClickGraph) outperform the XGBoost models in terms of AUC and accuracy. AUC tends to be a reliable overall measure as it summarizes the trade-offs between precision and recall. In the case of ClickGraph, the hidden layers are able to learn a representation of this intent in order to better identify users who will click "form fill."

Modeling the graph structure improves performance In our experiments we test two different variants of the classic XGBoost algorithm, as well as two variants on an LSTM. In both cases, graph information is incorporated in one, and not the other. Table 1 shows that for the XGBoost models, XGBoost (graph) outperform XGBoost (Naive) in terms of AUC, and accuracy. Additionally, we can make

Model	AUC	F1	Accuracy	Precision	Recall	FN rate
MajorityClass	0.5	0.0	0.9768	0.0	0.0	1.0
XGB (naive)	0.7989	0.2022	0.9725	0.3103	0.15	0.85
XGB (graph)	0.8592	0.0187	0.9756	0.1429	0.01	0.99
LSTM (no graph)	0.9388	0.0082	0.9812	**1.0**	0.0041	0.9959
ClickGraph	**0.9493**	**0.2913**	**0.9829**	0.6716	**0.1860**	**0.8140**

Table 1: Test set evaluation metrics for various classifiers. We observe that the proposed method, ClickGraph, outperforms all baselines along key metrics of interest: AUC, F1, Recall, and FN rate. For all relevant metrics, we use a classification threshold of .5

a similar observation between the LSTM-based models LSTM (no graph) and ClickGraph. The proposed model outperforms the more naive LSTM along a host of metrics: all but precision. While we do see obvious tradeoffs between FP's and FN's when analyzing the graph-based and graph-free models, the overall differences in AUC gives us insight into the nature of these tradeoffs.

Simultaneously predicting different actions reduces FN's and improves recall Note that the two models that yield the best AUC are the two recurrent neural networks, LSTM (no graph) and ClickGraph. While the AUC and accuracy metrics between the two are somewhat comparable, ClickGraph dramatically outperforms the naive LSTM in terms of F1, recall, and FN rate. This result seems to suggest that the multi-task aspect of the model, wherein we can encode the graph structure of the website through next page prediction, aids in reducing false negatives.

Additionally, it is worth noting the high precision of LSTM (no graph) as compared to ClickGraph. The perfect precision for LSTM (no graph) occurs because it predicts only one TP and zero FP's. In other words, the model is extremely biased towards $\hat{y} = 0$. In fact, it is hardly different the MajorityClass model. Conversely, ClickGraph dramatically increases TP's and decreases FN's. While it does incur more FP's, the cost of this error is low in the context of our problem. Additionally, overall model performance is maintained, demonstrated by high AUC and accuracy. As mentioned above, these results suggest that ClickGraph is a highly desirable model in the context of predicting form fill conversion.

6 CONCLUSION

In this paper, we studied user web behavior in a digital marketing context. We defined the new problem of simultaneously predicting multiple user actions from sequential, clickstream data. This multi-task approach to predicting future user actions is effective at improving prediction performance of high interest tasks, such as such as form fill conversion. We proposed a novel method called ClickGraph to fully utilize sequential, and graph structure in clickstream data. This same method can easily be generalized to predicting a variety of other user actions. We evaluated our method against strong baselines on real-world clickstream data. Our experiments demonstrate that ClickGraph is very effective at reducing false positives, and improving recall. We hope to explore other data sets and other settings, in particular different actions sets, for future work.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). https://www.tensorflow.org/ Software available from tensorflow.org.

[2] et al. Chantat Eksombatchai. 2018. Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time. *WWW 2018: The 2018 Web Conference, April 23-27, 2018, Lyon, France* (2018), 10.

[3] Ram Kosuru Choudur Lakshminarayan and Meichun Hsu. 2016. Modeling Complex Clickstream Data by Stochastic Models: Theory and Methods. In *WWW'16 Companion, April 11–15, 2016, Montreal, Quebec, Canada.* ACM, 7 Pages.

[4] Ewa Dominowska and Vanja Josifovski. 2008. First Workshop on Targeting and Ranking for Online Advertising. In *WWW'08 Companion, April 21–25, 2016 Beijing, China.* ACM, 2 Pages.

[5] Optimezely Editors. 2018. Landing Page Optimization. (2018). Retrieved November 3, 2018 from https://www.optimizely.com/optimization-glossary/landing-page-optimization

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* MIT Press. http://www.deeplearningbook.org.

[7] Sule Gunduz and M. Tamer Ozsu. 2003. A Web Page Prediction Model Based on Click-stream Tree Representation of User Behavior. In *Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining.* ACM, 6 Pages.

[8] Chengru Song Ying Fan Han Zhu Xiao Ma Yanghui Yan Junqi Jin Han Li Guorui Zhou, Xiaoqiang Zhu and Kun Gai. 2018. Deep Interest Network for Click-through Rate Prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining.* ACM, 10 Pages.

[9] Rami Al-Rfou Haochen Chen, Bryan Perozzi and Steven Skiena. 2018. A Tutorial on Network Embeddings. *arXiv preprint arXiv:1808.02590v1* (2018).

[10] Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation* (1997), 9(8):1735–1780.

[11] Jan Koutnik Bas R. Steunebrink Klaus Greff, Rupesh K. Srivastava and Jurgen Schmidhuber. 2015. LSTM: A Search Space Odyssey. *Transactions on Neural Networks and Learning Systems* (2015).

[12] Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective.* The MIT Press, Cambridge, Massachusetts.

[13] Jay Adams Paul Covington and Emre Sargin. 8. Deep Neural Networks for YouTube Recommendations. In *RecSys'16 September 15-19, Boston, MA, USA.* ACM, 10 Pages.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[15] 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 (KDD '14).* ACM, New York, NY, USA, 701–710. https://doi.org/10.1145/2623330.2623732

[16] Ruofei Zhang Zhongfei (Mark) Zhang Shuangfei Zhai, Keng-hao Chang. 2016. DeepIntent: Learning Attentions for Online Advertising with Recurrent Neural Networks. In *Proceedings of the 22th ACM SIGKDD international conference on knowledge discovery and data mining.* ACM, 10 Pages.

[17] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* (2009).