Multitask Mixture of Sequential Experts for User Activity Streams

Zhen Qin*, Yicheng Cheng*, Zhe Zhao, Zhe Chen, Donald Metzler, Jingzheng Qin
Google LLC, Mountain View, CA, USA
{zhenqin,chengyicheng,zhezhao,chenzhe,metzler,jzq}@google.com

ABSTRACT

It is often desirable to model multiple objectives in real-world web applications, such as user satisfaction and user engagement in recommender systems. Multi-task learning has become the standard approach for such applications recently.

While most of the multi-task recommendation model architectures proposed to date are focusing on using non-sequential input features (e.g., query and context), input data is often sequential in real-world web application scenarios. For example, user behavior streams, such as user search logs in search systems, are naturally a temporal sequence. Modeling user sequential behaviors as explicit sequential representations can empower the multi-task model to incorporate temporal dependencies, thus predicting future user behavior more accurately. Furthermore, user activity streams can come from heterogeneous data sources, such as user search logs and user browsing logs. They typically possess very different properties such as data sparsity and thus need careful treatment when being modeled jointly.

In this work, we study the challenging problem of how to model sequential user behavior in the neural multi-task learning settings. Our major contribution is a novel framework, Mixture of Sequential Experts (MoSE). It explicitly models sequential user behavior using Long Short-Term Memory (LSTM) in the state-of-art Multi-gate Mixture-of-Expert multi-task modeling framework. In experiments, we show the effectiveness of the MoSE architecture over seven alternative architectures on both synthetic and noisy real-world user data in G Suite. We also demonstrate the effectiveness and flexibility of the MoSE architecture in a real-world decision making engine in GMail that involves millions of users, balancing between search quality and resource costs.

KEYWORDS

multi-task learning, LSTM, activity stream

ACM Reference Format:
Zhen Qin*, Yicheng Cheng*, Zhe Zhao, Zhe Chen, Donald Metzler, Jingzheng Qin. 2020. Multitask Mixture of Sequential Experts for User Activity Streams. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403359

*Equal contribution.

This work is licensed under a Creative Commons Attribution International 4.0 License.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA
© 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7998-4/20/08.
https://doi.org/10.1145/3394486.3403359

1 INTRODUCTION

It is often desirable to model multiple objectives simultaneously in many web applications [5, 9, 36]. For example, it is beneficial to optimize user engagement (such as click) and satification (such as rating) for recommender systems [36]. Recently, neural multi-task learning, which jointly models multiple objectives, has been actively researched for such kind of problems and has been deployed in several real-world large-scale commercial web systems. For example, a large scale multi-objective ranking system was introduced for recommending the next video to watch on an industrial video sharing platform, and the system was designed to optimize for multiple objectives, such as user satisfaction and user engagement [42].

Multi-task learning is effective especially when tasks are closely correlated [24]. First, it allows efficient knowledge and data sharing across relevant tasks. This potentially improves performance of all tasks involved, especially those with sparse signals when tackled alone (e.g. conversion in e-commenrce applications [25]). Also, multi-task learning can act as a regularizer by introducing an inductive bias [32], so auxiliary tasks can be used to improve the generalization of the main tasks.

While most of the multi-task model architectures proposed to date focus on using non-sequential input candidate features (e.g., query and context) [32], input data is often sequential in real-world data science application scenarios. For example, web documents are sequential: they often consist of a sequence of words, and the state-of-art machine translation models usually explicitly model the sequential nature of the data using LSTM cells [37] or attentions [34]. Another example, which we focus on in this work, is the ubiquitous user activity data, which plays a key role in personalization-oriented applications such as recommender systems and personal assistants [33]. A user activity stream describes a user's sequential behavior. Effective sequential modeling leads to better user behavior prediction for future decision making needs such as recommending a relevant item [6]. Modeling sequential user behaviors as explicit sequential representations can empower the multi-task model to incorporate temporal dependencies to achieve better performance.

In this paper, we study the problem of multi-task learning when the model consumes sequential user activity data. While multi-task learning can potentially help to learn better joint representations for different user behavior objectives, we face the following challenges:

- **Data Sparsity** — User activities can be highly sparse in real-world web applications. For example, user purchase events can be very rare compared with other events, such as user impression events.
- **Data Heterogeneity** — User activity data is heterogeneous and spans a variety of data sources and types. For example, user profile data contains gender information, while user log data contains clickthrough information. Learning shared

representations for those heterogeneous data is known to be difficult due to task conflicts [35, 42].
- **Complex Multiple Objectives** — The (temporal) relationship among objectives, such as click and purchase, can be complicated due to a user's complex underlying intent [36].

Our major contribution in this work is a model called Mixture of Sequential Experts (MoSE) that addresses the above challenges. The model is a novel combination of the state-of-art multi-gate mixture of experts (MMoE) multi-task learning model [24] and Long Short-Term Memory (LSTM) [15]. Based on both controlled synthetic dataset and a real-world dataset that involves millions of users in G Suite (i.e., GMail and Google Drive), we propose and explore several principled ways to model multiple objectives in user activity streams.

Our findings from the experiments include: 1) We show the benefits of MoSE on both the synthetic dataset and a real-world decision making engine in GMail that involves millions of users. MoSE consistently outperforms seven other alternatives by a large margin on all the tasks, including predicting user keypress and mouse clicks in G Suite.

2) We validate the design choices of MoSE through an ablation study. We find that it is the combination of sequential representation and MMoE that allows the effective modeling of user activity stream data. MMoE alone fails to explore the rich sequential dependencies in user activity streams, and LSTM combined with standard multi-task algorithms fails to effectively model sparse and noisy variables as well as the interactions among them.

3) We highlight the flexibility of using MoSE in practice. It alleviates the need of task training weight tuning for achieving accurate predictions on all tasks. Due to the accurate modeling of MoSE, it outperforms baselines consistently on serving with different business needs *after* a single MoSE model is trained.

To summarize, our main contributions in this work are:

- We study the important but under-explored problem of modeling multiple objectives in user activity streams.
- We propose a novel Mixture of Sequential Expert (MoSE) model that consistently outperforms alternatives while providing important practical flexibility in both a synthetic and a real-world large-scale user activity dataset.
- We perform the ablation study to show the benefit and necessity of the design choices of MoSE.
- We show a successful application of MoSE in a real-world multitask decision making service, highlighting the model's effectiveness and flexibility.

2 RELATED WORK

2.1 User activity stream modeling

Recently, using neural sequence models to effectively represent users' activity streams has become popular in web applications. The work in [3] studies how to effectively use context features (such as device) for recommending videos in Youtube. It uses LSTM to represent a user's video watch history. [30] is a personalized sequential recommendation model based on Markov chains. [39] applies a Recurrent Neural Network (RNN) for next basket recommendation. [33] uses RNN and Attention to model both short and long range dependencies in user sequences. [17] proposes a self-attention based sequential model for next item recommendation [13] shows that RNN can learn multiple user dynamics patterns in individual recommendation sessions. These works show the benefit of sequence modeling over non-sequential models (e.g., fully connected feed-forward neural networks). However, none of these works studied the problem of modeling multiple objectives in their setting. Also, existing work typically focus on a single data source (e.g., video watch history), while in practice it may be beneficial to model streams from multiple data sources for a more holistic view of the users. Heterogeneous data sources possess different properties such as data sparsity, and we show it is critical to use dedicated components for objectives from different sources using a mixture of experts approach.

2.2 Multi-objective Optimization (MOO) for e-commerce and recommender systems

It is desirable to model multiple objectives in real-world applications, such as optimizing click-through rate and gross merchandise volume in recommender systems [20], clicks and purchases in E-Commerce searching and advertising [25, 36]. Recently, neural multi-task learning, which jointly models multiple objectives, has been actively researched for such kind of problems[32, 40].

A family of approaches combine multiple objectives into a single objective before model learning, in which the combined weights are usually selected heuristically [31]. The recent work [5] shows it can be beneficial to stochastically aggregate the labels. Another recent work [20] is a theoretically driven approach to learn the optimal weighting parameters in the context of Pareto efficiency (one objective can not be optimized without hurting other objectives). The focus of such work is around the objectives (such as how to aggregate labels). Our work is complimentary to them and mainly focuses on multi-task deep learning with flexible parameter sharing, discussed next.

2.3 Multitask deep learning with flexible parameter sharing

A hot research topic in the deep learning domain is to design model architecture to specifically facilitate the modeling of multiple objectives. Multi-gate Mixture-of-Expert [24] is one of the state-of-the-art in multi-task modeling, showing better results in content recommendation over several competing approaches including tensor factorisation [38] and a cross-stitch network [27]. Some work [8, 11, 41] explicitly optimized the multiple objectives by considering the task relationships. However, those works only considered non-sequential input data, while in this paper, we focus on modeling the sequential inputs from user activity streams.

[25, 36] focuses on parameter sharing and optimization schemes that allow better transfer learning from clicks to the more sparse conversions. This family of work usually has one major task and other tasks are considered auxiliary. Our work focus on general methods for optimizing multiple objectives simultaneously. Also, none of these works provide a complete treatment that takes advantage of the sequential nature of user activity data.

2.4 Sequential and structural multi-task learning

Neural multi-task learning with sequential input data was studied recently for natural language processing (NLP) tasks [10, 16, 22]. These studies focus on tasks such as machine translation where input data is single dimension sequences, i.e., text sentences. Multi-task modeling is also successful in many computer vision tasks [4, 21], where the input is homogeneous images or videos. Different from this line of research, in this work, we focus on modeling user activity streams from heterogeneous data sources (e.g., search logs and browsing logs) and the interactions among them.

3 A MOTIVATING APPLICATION

In this section, we show a motivating real-world decision making service in GMail that involves millions of users, where modeling multiple objectives in user activity streams is needed.

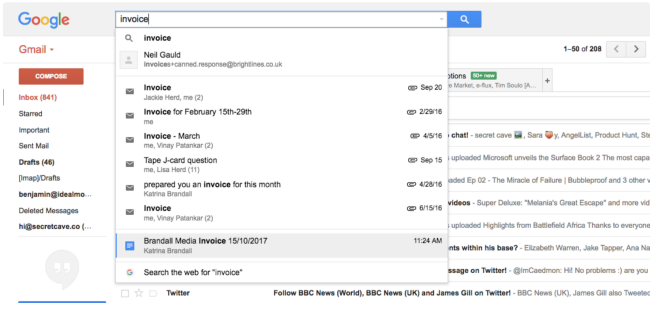


Figure 1: GMail search allows users to search both emails and Drive documents [1].

GMail is an email service and Google Drive is a file sharing and storage system. As shown in Figure 1, when a user searches in the GMail search UI, Drive documents may show up when available. This requires a search request from GMail to the Drive search backend system. Furthermore, GMail search enables a "search-as-you-type" feature, where each *keypress* in the search box triggers a search request. The large GMail search volume causes a highly non-trivial resource burden to the Drive search system. On the other hand, document search in GMail has relatively low utility, while many users find this feature to be useful. Thus, instead of completely turning this feature off, machine learning models can be built to selectively turn on this feature at an individual level. In practice, the decision of whether or not to suppress the section is refreshed daily.

This problem requires the modeling and prediction of the two tasks in the G Suite data: the number of Drive search result clicks and the number of keypresses when a user searches in GMail. The latter is a proxy for resource cost due to the "search-as-you-type" feature. Making the decision to turn on the document search feature is a balance between the two tasks. Ideally we want to turn off the feature for users who will perform a lot of keypresses but few document search clicks. Furthermore, the balancing factor between these two tasks may change due to business needs, such as the search request capacity the Drive search backend allows. Ideally, we do not want to re-train the model every time the business needs change. The model predictions need to be accurate and robust under these requirements.

This problem setting is common in industrial applications where a personalized decision can be made to trade-off multiple user-facing or non-user facing objectives. It possesses several challenges in user activity stream modeling. First, many variables, such as one of the tasks, Drive search result click, is highly sparse. Second, besides GMail activity, Drive activity (e.g., document edit, open, and creation) needs to be considered (intuitively, how user behaves in Drive can affect how she reacts to a Drive document in GMail). so the problem requires modeling data from multiple data sources which can contain heterogeneous data, including search logs in GMail, browsing logs in GMail, and activity logs in Google Drive. Third, the objectives are complex, since how user clicks and press keys can depend on a user's complex underlying intent. As we will show, standard non-sequential multi-task models and straightforward sequential extensions of multi-task models do not work well under these challenges.

4 METHOD

We first discuss the preliminaries for modeling multiple objectives in user activity stream data in Section 4.1. Then we introduce the Mixture-of-Sequential-Experts (MoSE) model in Section 4.2, and alternatives for modeling user activity stream data in Section 4.3.

4.1 Modeling Preliminary

We can represent a user activity stream formally as an (N, T, D) tensor, where N is the number of samples (e.g. one sample per user), T is the length of the sequence or the number of time steps, and D is the dimension of variables we care about. Each sample $x = [\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, ...\mathbf{x}_{(T)}]$, where $\mathbf{x}$ is a D dimensional vector. Each variable is associated with one user activity event type and may possess sparse or dense values. For example, a real-valued variable $x^i_{(t)}$ may describe the number of clicks for a user at time t. This representation generalizes session-based recommendation datasets where $x^i_{(t)}$ can be a sparse vector describing which item a user clicked and $x^j_{(t)}$ is a sparse vector describing which item a user purchased at time t.

We are interested in modeling a subset of the D dimensional variables that are of interest (e.g. business related) as the task variables in a multi-task learning setting. Our objective is to predict the task variable values for $t > T$, which can be used for downstream applications such as item recommendation.

4.2 Mixture of Sequential Experts model

In this section, we propose the Mixture of Sequential Experts (MoSE) framework for multi-task modeling of sequential user activity data.

The sequential multi-task learning setting presents us with unique challenges. First, user activity data (e.g. user profile and click-through information) can be sparse and heterogeneous. For example, the available user click-through information varies a lot among different users (e.g. active vs. inactive users) and modeling user behaviors in one data source can be very different from another one (e.g. the same user's behavior in GMail vs Google Drive). In addition, the temporal relationship among multiple objectives

can be complicated due to a user's complex underlying intent. We thus hypothesize that, in addition to an explicit sequential approach for data representation, there should be dedicated components in the framework that model different aspects of the complex dataset before merging them.

As a result, we propose the MoSE framework to address the above issues. As shown in the Figure 3, MoSE is composed of the following components.

- A shared-bottom LSTM module for consuming sequential input data, which allows explicit and effective representation learning from the input layer.
- A mixture of sequential experts layer where each expert models different aspects for each task. For example, one expert can focus on modeling the sequential dependency of a sparse variable. The main goal of the Mixture of Expert (MoE) layer [12] is to achieve conditional computation, where only parts of a network are active on a per-example basis. We further augment the MoE layer by using LSTM instead of fully connected net to better handle sequence data.
- Gating networks to gate the outputs of experts as proposed in [24]. Each gating network can learn to "select" a subset of experts to use conditioned on the input example. This allows the modeling of complex interactions among heterogeneous variables.
- Multi-tower network with one tower per task to decouple the optimization for tasks. This is a common structure in the multi-task learning literature which is known to be useful for learning different tasks with varying scales and data types.

To summarize, MoSE provides a full sequential solution with gated mixture-of-experts to model user activity streams. In addition to the natural fit of LSTM for sequential data, recent research shows that LSTMs are efficient to learn from sparse event data [26] that is common in user activity streams. The mixture of experts framework allows dedicated sequential experts to focus on different challenging aspects of the data, such as modeling variables that are sparse or have complex temporal dependencies. The gated mixture of experts [24] module allows each task to pick the most relevant experts. This is important for modeling user activity data where the task relationships may be weak or difficult to learn. On the other hand, traditional multi-task frameworks that use a single shared component can get confused by heterogeneous data sources with dramatically different characteristics, such as sparsity and noisiness.

Mathematically, given a data sequence with T time steps $x = \{\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, ..., \mathbf{x}_{(T)}\}$, we perform many-to-many sequence learning (see Fig. 2). For a given time step t with input $\mathbf{x}_{(t)}$, we can formulate the output for task k at time $t+1$ as below:

$$
\begin{aligned}
y^k_{(t+1)} &= h^k_{LSTM}\left(f^k\left(\mathbf{x}_{(t)}\right)\right) \\
\text{where} \quad f^k\left(\mathbf{x}_{(t)}\right) &= \sum_{i=1}^{n} g^k(x)_i \, f_{LSTM_i}\left(f_{LSTM}\left(\mathbf{x}_{(t)}\right)\right)
\end{aligned}
\tag{1}
$$

where h^k_{LSTM} is the tower network for task k, f^k is the output of the gated mixture of experts layer, f_{LSTM_i} is the i-th sequential expert (there are n experts in total), f_{LSTM} is the shared bottom

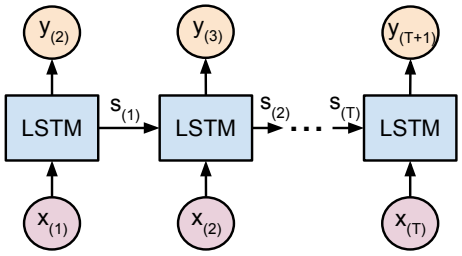


Figure 2: The many to many LSTM structure.

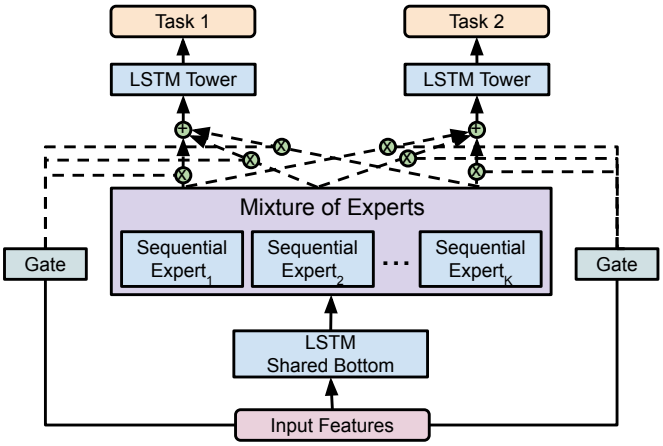


Figure 3: Illustration for the MoSE model structure. Note that we use two tasks for simple illustration but the framework allows more objectives.

network, g^k is the gating network which transforms the input to a distribution over the n experts based on the input:

$$
g^k(x) = softmax(W_{gk}x) \tag{2}
$$

where $W_{gk} \in R^{n \times DT}$ is the weight matrix we are trying to optimize. Note that we calculate the gating distribution on the whole sequence x rather than at each time step. We are doing this to get a consistent gating distribution at any time step for a specific data sample. In equation 1, functions with $_{LSTM}$ subscripts can be stacked LSTM networks instead of a single layer.

4.2.1 Discussion. We also experimented with gated recurrent unit (GRU [7]) and regular recurrent units as the building block for the sequential units. We found the performance were inferior or similar to LSTM in general and omit their results due to space constraints. Experimenting with more advanced techniques such as Transformer [34] is considered as future work.

4.3 Alternatives

To show the advantage of MoSE for multi-task modeling in user activity data, we evaluate seven alternative approaches.

4.3.1 Separate Model for Each Task (Multi-Model). We can build a standalone neural network for each task for comparison to see

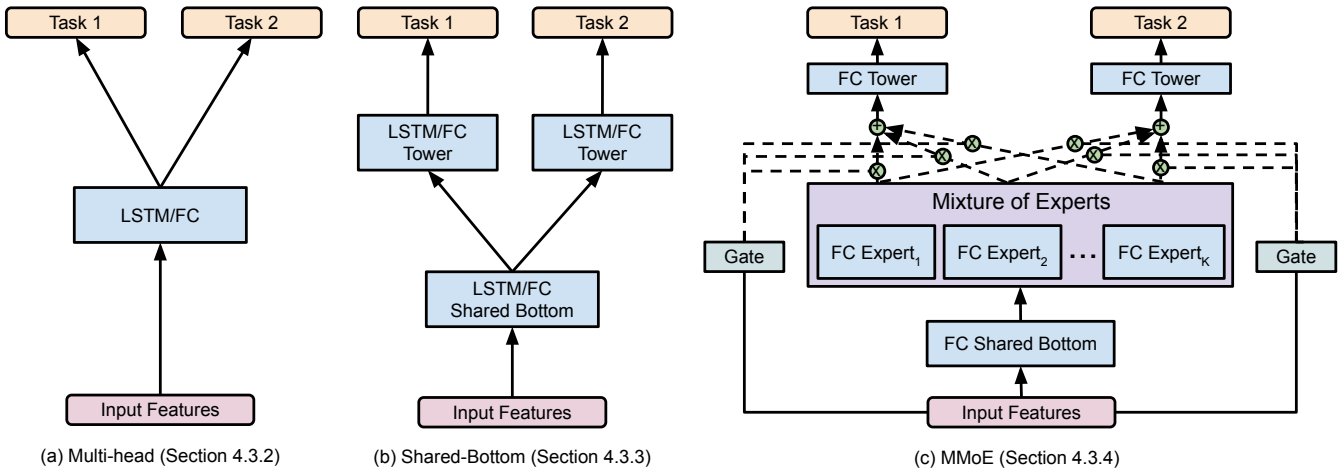


Figure 4: Different model structures for the alternative multi-task models we consider. Note that we use two tasks for illustration but the frameworks allow more tasks. FC stands for fully-connected layers.

how multi-task approaches perform in general. Due to the subtlety of making multi-task approaches work (e.g., some methods assume strong correlation between tasks), it should not be surprising if some multi-task approaches perform worse than this approach. On the other hand, this approach will not benefit from multi-task learning, such as knowledge sharing, and may require extra effort to maintain multiple models instead of one joint model. We build two variants: *Multi-Model* uses a fully-connected feed-forward network for each task and treats the input tensor as non-sequential data. *Sequential Multi-Model* uses stacked LSTMs to ingest sequential data.

4.3.2 Multi-head model (Multi-head). This approach first predicts the task objectives and merges the losses into a single one before back propagation. This is the standard approach for predicting multiple outputs. Merging the losses without dedicated components such as separate towers for each task may not perform well for conflicting tasks [32]. We also evaluate two variants as before: *Multi-head* that uses a fully-connected feed-forward network and *Sequential Multi-head* that uses stacked LSTMs.

4.3.3 Shared-bottom Model (Shared-Bottom). This approach is one of the most common multi-task learning approach [32]. It uses multiple objectives (or heads) and each task has its own tower after the shared-bottom module. The loss for each individual task is first calculated and then only combined before back propagation. The model allows both knowledge sharing (via shared-bottom) and specificity (via individual towers). However, the dedicated specificity components are close to the final output and may not handle input heterogeneous data sources well early on. Similar as before, we build *Shared-bottom* and *Sequential Shared-bottom*.

4.3.4 Multi-gate Mixture-of-Experts (MMoE). The Multi-gate Mixture-of-Experts model [24] is the state-of-art deep multi-task learning approach. It can automatically adjust parameterization between modeling shared information and modeling task-specific information. Experiments on several non-sequential datasets show that it works particularly well on tasks that are less correlated. It is not clear how MMoE works on sparse and noisy user activity data. MoSE is an extension of MMoE and we will show that it is the combination of MMoE and LSTM that can handle the various challenges of user activity data, while MMoE itself does not show strong benefits over some other non-sequential models.

We show the model architecture for the different multi-task alternatives in Figure 4.

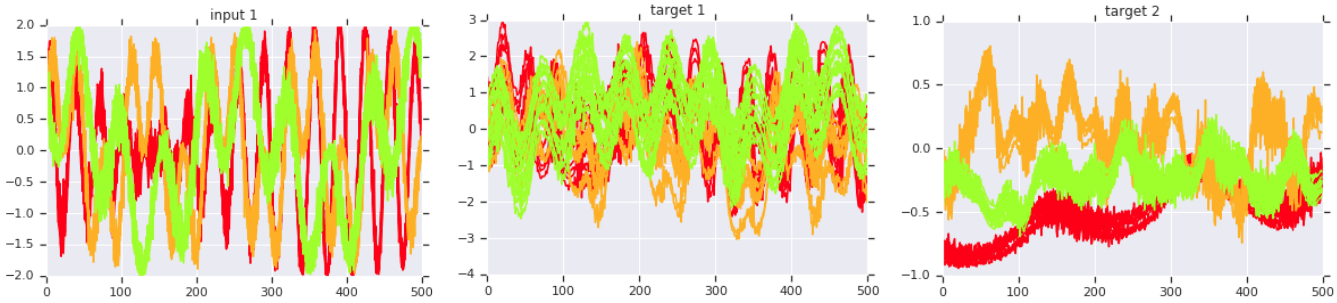
5 EVALUATION

We validate the effectiveness of MoSE on a synthetic sequential multitask dataset and a real-world user activity stream dataset in G Suite. We also show the application of MoSE in a decision making service where multiple objectives are traded-off due to business needs.

5.1 Synthetic Experiments

In order to validate how MoSE and the alternatives perform on general sequential multi-task applications, we design synthetic experiments extending the data generation method in [24].

5.1.1 Dataset. We generate a sequential synthetic dataset using a mixture of sinusoidal functions shown in Table 1, and give an illustration of the data in Figure 5. We generate D-dimensional data with multiple modes. For each data point, we first choose the mode m it belongs to and generate the input by feeding a continuous time stamp t to $in(t)$. We then take $in(t)$ as input, calculate $out_1(t)$ and $out_2(t)$ with their parameters $v_1^{(m)}$ and $\{v_2^{(m)}, b_2^{(m)}\}$ correspondingly. Thus, $in(t)$ is a mixture of O sine waves with mixing weight $w_o^{(m)}$, which is a length D vector used for weighting of the D dimensional input; $out_1(t)$ is a mixture of D sine waves which sum all dimensions of the weighted input from $in(t)$ at each time t with weight $v_{1d}^{(m)}$ for dimension d; $out_2(t)$ is sine of weighted sum over all dimensions of input from $in(t)$ with weight $v_{2d}^{(m)}$ and bias $b_{2d}^{(m)}$; ϵ is the random noise added to all phases when generate the data. This dataset is sufficiently complex for a synthetic experiments,



(a) Input of the synthetic data. (b) First target of synthetic data. (c) Second target of synthetic data.

Figure 5: Illustration for the synthetic dataset. The horizontal axis is time, vertical axis is the feature or target value. In order to add complexity to the dataset we generate the data using multiple modes of parameters which are shown by different colors.

where the two tasks are correlated due to the shared calculations (e.g., the sine operation).

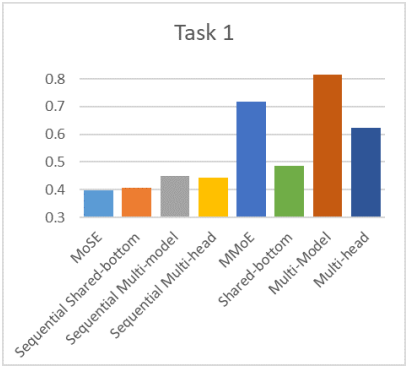
Table 1: Formulas for generating the synthetic data.

Input $in(t)$	$\sum_o sin\left(w_o^{(m)} t + \epsilon\right)$
Target 1 $out_1(t)$	$\sum_d sin\left(in\,(t)\, v_{1d}^{(m)} + \epsilon\right)$
Target 2 $out_2(t)$	$sin\left(\sum_d \left(\frac{in(t) v_{2d}^{(m)}}{D} + b_{2d}^{(m)} + \epsilon\right)\right)$

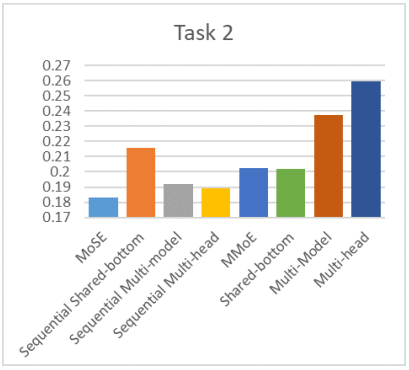
We generated a synthetic dataset of 2000 data points as described above with $M = 10$, $D = 10$, $O = 2$, $t = 0.2i$ where i is an integer ranging from 0 to 500. Then we take $in(t)$, which is a 2000 by 500 by 10 tensor as the input data. We call $out_1(t)$ target 1 and $out_2(t)$ target 2. Both of the target 1 and target 2 are 2000 by 500 tensors.

5.1.2 Experiment settings. We use 80% of the samples for training, 10% for validation, and 10% for testing. Mean Squared Error (MSE) is calculated between prediction and ground truth for the last 7 time steps for the loss and evaluation metric. Non-sequential models output a 7-dimension vector for each task. All the models are implemented using the Tensorflow toolkit [2] and optimized with the Adam optimizer [18]. For all models we pick the hyperparameters using cross-validation by varying the number of network layers from 1 to 3 and number of neurons in each layer ranging in [16, 32, 64, 128, 256] to decide the best network structure for both LSTM and fully connected modules, including the shared bottom, experts, and task towers when applicable. The importance weights between two tasks is set to 1 when applicable (only Multi-model does not need this) since the target tasks are of the same scale and we do not assume prior knowledge of the importance weights. The best network structure learned for MoSE is a single layer LSTM with 16 units for the shared bottom, single layer LSTM with 8 units for each of the 10 experts, and 8 LSTM units for the towers.

5.1.3 Results. We summarize the results on the test dataset in Figure 6. The Figures show the Mean Squared Error of MoSE comparing to the seven alternative methods when predicting the two tasks on the synthetic dataset. We can see that MoSE achieves the best performance consistently for both tasks. When compared with the second best model (Sequential Multi-head), MoSE produces a roughly relative 10% smaller error. We defer more detailed discussions to the next section.



(a) Average MSE for task 1.



(b) Average MSE for task 2.

Figure 6: Model performance on the synthetic dataset.

5.2 Experiments on G Suite data

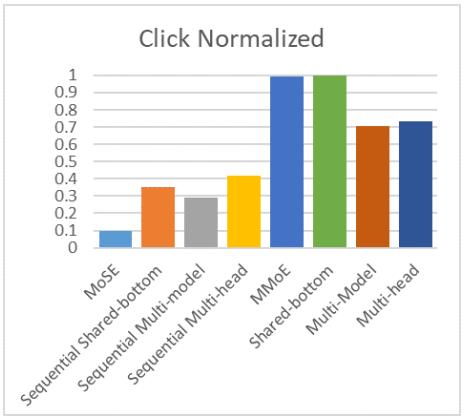
5.2.1 Dataset. We collect a sample of G Suite data, which contains user activity data from heterogeneous sources in G Suite including

GMail and Google Drive. The dataset contains nearly 10 million data points in total across 30 days of user activity logs. We take each day as one time step. Each variable represents one user's activity. The set of variables we consider include Google Drive activity counts, including the numbers of Drive document views, Drive document edits, Drive searches, as well as GMail search behavior statistics, including the number of key-presses when a user searches, email search result clicks, and Drive search result clicks. See Figure 1 for an illustration of search both email and Drive documents in GMail. Two of these variables in GMail are our tasks, i.e. number of key-presses when a user searches, and the number of Drive search result clicks. We will show an application that leverages these task predictions in the next section. The dataset is quite sparse, especially for one of our tasks - the number of Drive search result clicks in GMail. When no relevant activity is performed in a day, the corresponding fields are filled with zero.

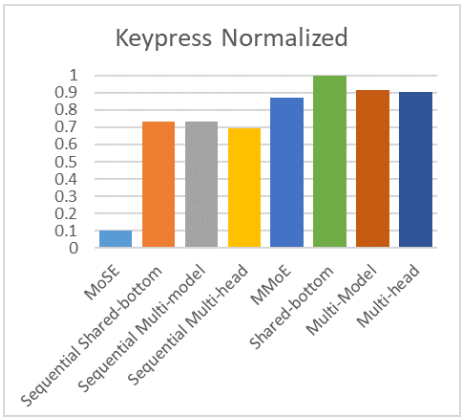
5.2.2 Experiment settings. Due to the sparsity of the user data, we first sub-sample the training data to prevent the model from over-fitting on zeros. First, we removed users with no activities in both of the two tasks and then sub-sampled the data to ensure there are at least 20% of users in the training data that have click task activity in the 30-day period. Notice that the sub-sampling is applied only on the training data. For evaluation we are using all the test data. We randomly sampled 80% data as training, 10% for validation, and the remaining 10% for evaluation. For each data sequence (user), we predict the target tasks in the last day using information before that, and calculate the Mean Squared Error (MSE) between the predictions and observed values. The model hyper-parameters are selected in the same way as Section 5.1 and the important weights between tasks is also set to 1 when applicable for all models for fair comparison. The best network structure learned for MoSE is a two layer LSTM with $[128, 64]$ units for the shared bottom, two layer LSTM with $[64, 64]$ units for each of the 10 experts, and two layer LSTM with $[64, 32]$ units for the towers.

5.2.3 Results. We again compare MoSE with seven alternative models and show the results for the G Suite data in Figure 7. We show relative performance of the models due to the sensitivity of the data. Combined with the results from Section 5.1, we can make the following observations:

- MoSE consistently outperforms the alternatives, especially on the complex real-world dataset.
- Sequential models in general outperform non-sequential models, showing the necessity of explicitly modeling sequential dependencies in user activity streams.
- MoSE significantly outperforms other sequential models. This shows the mixture of sequential experts module is able to effectively handle various challenges in user activity streams such as sparse variables and complicated interactions between heterogeneous data sources.
- MMoE itself does not show significant benefits over other non-sequential models. The mixture-of-experts framework is most beneficial when we use sequential experts as in MoSE since most complexity in user activity streams seem to stem from sequential complexity and sparsity.



(a) Relative average MSE of click prediction.



(b) Relative average MSE of keypress prediction.

Figure 7: Model performance on G Suite data.

5.3 Trading-off resource cost and document search in GMail search

In this section, we show the application of MoSE in the real-world decision making service in GMail described in 3. It requires the modeling of the two tasks in the G Suite data discussed in Sec 5.2: the number of Drive search result clicks and the number of keypresses when a user searches in GMail.

5.3.1 Experiment setup. We compare MoSE trained in Section 5.2 with the production model in the GMail search system. The production model is a heavily tuned Shared-Bottom model with the goal to save 80% resource compared with always turning on the document search feature in GMail.

After obtaining the prediction for key-presses K_p and click C_p, we decide whether to turn on or off the document search feature based on the weighting parameter α (we differentiate between α and r to highlight that α is used for inference and r is used in training) and the decision threshold θ by the decision label $L_p = (C_p - \alpha K_p) > \theta$. If the decision label L_p is true for a certain user, we turn the feature on and turn it off for users with L_p = false. With

the decision label L_p we can define the resource saving rate P_r and click preserving rate P_c as below:

$$P_r = 1 - \frac{\sum_i (1 - L_{p_i}) * K_{gt_i}}{\sum_i K_{gt_i}} \tag{3}$$

$$P_c = \frac{\sum_i L_{p_i} * C_{gt_i}}{\sum_i C_{gt_i}} \tag{4}$$

where $\sum_i$ is the sum over all users, C_{gt_i} is the ground truth number of Drive search result clicks for user i, K_{gt_i} is the ground-truth number of key-presses for user i, and L_{p_i} is the decision label described above for user i. We can easily see from the formula that if we set the decision threshold θ to the minimum of $C_{p_i} - \alpha K_{p_i}$, we will turn on the document search feature for all users. This will result in a click preserving rate 1 with no resource savings. On contrary, if we set θ to the maximum of $C_{p_i} - \alpha K_{p_i}$ we will have a L_p of all false, and hence we will turn off the document search feature for all users.

In order to evaluate how the models performed at different trade-off points, we measure the overall performance by the AUC (Area Under the Curve) score of the resource savings rate versus click preserving rate curve by varying θ and α. We perform a comprehensive grid search of θ and α on the validation set to find the best click preserving result at each resource saving level for the compared models. We report at different resource saving levels, the relative performance of click preserving of the two models on test set in Figure 8.

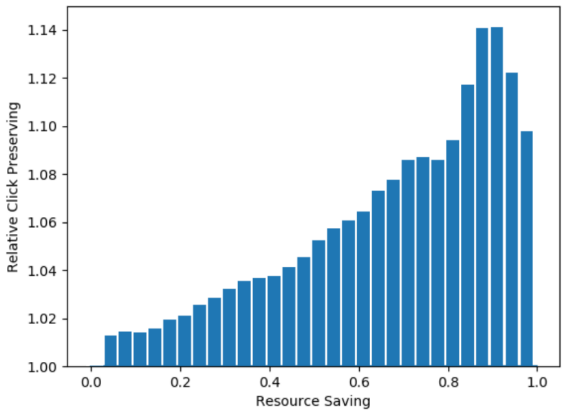


Figure 8: Relative click preserving of MoSE over the Shared-Bottom production model with different resource savings.

5.3.2 Results. In additional to the comparative results shown in Figure 8, MoSE achieves +4.8% AUC score than the production model. We emphasize two benefits of MoSE. First, performance-wise, MoSE significantly outperforms the heavily tuned shared-bottom model. At the requirement of 80% resource savings, MoSE is able to preserve approximately 8% more document search clicks, which is very significant in the product. Also, MoSE is robust across different resource saving level due to the its modeling power, even though we assigned equal weights to the tasks during training. This gives MoSE more flexibility when the business requirement keeps changing in practice since a more robust model like MoSE may alleviate the need to re-train the model, comparing with models that are more sensitive to the importance weights during training.

6 DISCUSSION AND FUTURE WORK

In this section, we discuss a few insights and limitations which we have learned from developing MoSE for modeling user activity streams.

6.1 Difficulty of multi-task user activity sequence modeling

Though recently sequential multi-task learning has been explored for NLP tasks such as machine translation, where the tasks could be different language pairs, their inputs are limited to homogeneous, complete sentences [29]. In real-world applications, user activity streams possess unique difficulties that make existing work ineffective. User activities are typically very noisy and sparse, and come from multiple heterogeneous data sources due to the large amount of logging systems around users. Techniques that center around self-supervised learning [10] may not work well due to the sparsity of data. In this work, we focus on the modeling architecture to address the difficulties, including explicit sequential models and dedicated components to model different aspects of the data. But the problem setting opens opportunities to more thorough studies of user behavior across different applications, and more explicit handling of user's complex latent intents.

6.2 Extensibility of MoSE

We note that when we develop MoSE, we focus on the general architecture that tries to address the difficulties in real-world user activity streams. MoSE, consisting of general building blocks, can be easily extended, such as using other sequential modeling units besides LSTM, including GRUs, attentions, and Transformers [34] for the shared bottom, sequential experts, and task towers. A more thorough study of using different sequential units and their combinations would be interesting.

6.3 Limitations of our work

In this work, though our dataset comes from multiple data sources with varying properties, we did not explicitly handle multi-model data [42] such as images or natural language inputs. Also, explicitly modeling context features (e.g., location of the user) is a hot topic recently [3, 28]. It would be interesting to extend MoSE to better handle such kind of data.

Our work is limited to studying two tasks due to our application needs. We plan to study how MoSE scales up to more tasks.

6.4 Future modeling work

Due to the recent popularity of multi-task learning, we plan to integrate MoSE with more multi-task modeling techniques. For example, the MoSE architecture still has a LSTM shared-bottom component, which can be improved by even more flexible sharing. Novel model architectures such as Sub-network Routing [23] can

introduce more flexible parameter sharing and robust learned routing. Incorporating a causal objective when working with biased activity data [19] is also an interesting direction.

Our application only requires an offline inference of the UI decision for users daily. If the application requires an efficient online inference, the trade-off between effectiveness and efficiency should be considered. Exploring techniques such as model distillation [14] is a future direction.

7 CONCLUSION

In this work, we study the important but under-explored problem of learning multiple objectives in user activity streams. We propose a novel mixture of sequential experts (MoSE) framrwork that consistently outperforms alternatives on both synthetic and noisy real-world user activity data in G Suite. We further show an application of MoSE in a decision making service in GMail that affects millions of users.

REFERENCES

[1] 2014. 32 Google Drive Tips You've Probably Never Heard Before. https://www.process.st/25-google-drive-tips-youve-probably-never-heard-before/. Accessed: 2020-02-01.

[2] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*. 265–283.

[3] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *WSDM*. 46–54.

[4] Jiajiong Cao, Yingming Li, and Zhongfei Zhang. 2018. Partially Shared Multi-task Convolutional Neural Network with Local Constraint for Face Attribute Learning. In *CVPR*. 4290–4299.

[5] David Carmel, Elad Haramaty, Arnon Lazerson, and Liane Lewin-Eytan. 2020. Multi-objective Ranking Optimization for Product Search Using Stochastic Label Aggregation. In *WWW*.

[6] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*. 108–116.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv:1412.3555

[8] Carlo Ciliberto, Alessandro Rudi, Lorenzo Rosasco, and Massimiliano Pontil. 2017. Consistent Multitask Learning with Nonlinear Output Relations. In *NeurIPs*. 1983–1993.

[9] Onkar Dalal, Srinivasan H. Sengenmedu, and Subhajit Sanyal. 2012. Multi-objective Ranking of Comments on Web. In *WWW*.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *ACL-HLT*. 4171–4186.

[11] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser. In *ACL*. 845–850.

[12] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2013. Learning factored representations in a deep mixture of experts. arXiv:1312.4314

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. arXiv:1511.06939

[14] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*. http://arxiv.org/abs/1503.02531

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* (1997), 1735–1780.

[16] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics* (2017), 339–351.

[17] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. 197–206.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980

[19] Ang Li, Suming J. Chen, Jingzheng Qin, and Zhen Qin. 2020. Training Machine Learning Models With Causal Logic. In *WWW Companion*. 557–561.

[20] Xiao Lin, Hongjie Chen, Changhua Pei, Fei Sun, Xuanji Xiao, Hanxiao Sun, Yongfeng Zhang, Peng Jiang, and Wenwu Ou. 2019. A Pareto-Efficient Algorithm for Multiple Objective Optimization in E-Commerce Recommendation. In *RecSys*. 1–9.

[21] Shikun Liu, Edward Johns, and Andrew J Davison. 2019. End-to-end multi-task learning with attention. In *CVPR*. 1871–1880.

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

[23] Jiaqi Ma, Zhe Zhao, Jilin Chen, Ang Li, Lichan Hong, and Ed H. Chi. 2019. SNR: Sub-Network Routing for Flexible Parameter Sharing in Multi-task Learning. In *AAAI*.

[24] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In *KDD*. 1930–1939.

[25] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *SIGIR*. 1137–1140.

[26] Hongyuan Mei and Jason M Eisner. 2017. The neural hawkes process: A neurally self-modulating multivariate point process. In *NeurIPs*. 6754–6764.

[27] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *CVPR*. 3994–4003.

[28] Zhen Qin, Zhongliang Li, Michael Bendersky, and Donald Metzler. 2020. Matching Cross Network for Learning to Rank in Personal Search. In *WWW*. 2835–2841.

[29] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683* (2019).

[30] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. 811–820.

[31] Marco Tulio Ribeiro, Nivio Ziviani, Edleno Silva De Moura, Itamar Hata, Anisio Lacerda, and Adriano Veloso. 2014. Multiobjective Pareto-Efficient Approaches for Recommender Systems. *ACM Trans. Intell. Syst. Technol.* (2014), 53:1–53:20.

[32] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. arXiv:1706.05098

[33] Jiaxi Tang, Francois Belletti, Sagar Jain, Minmin Chen, Alex Beutel, Can Xu, and Ed H. Chi. 2019. Towards Neural Mixture Recommender for Long Range Dependent User Sequences. In *WWW*. 1782–1793.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762

[35] Ruoxi Wang, Zhe Zhao, Xinyang Yi, Ji Yang, Derek Zhiyuan Cheng, Lichan Hong, Steve Tjoa, Jieqi Kang, Evan Ettinger, and H Chi. 2019. Improving Relevance Prediction with Transfer Learning in Large-scale Retrieval Systems. (2019).

[36] Liang Wu, Diane Hu, Liangjie Hong, and Huan Liu. 2018. Turning Clicks into Purchases: Revenue Optimization for Product Search in E-Commerce. In *SIGIR*. 365–374.

[37] Yonghui Wu et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv:1609.08144

[38] Yongxin Yang and Timothy Hospedales. 2016. Deep multi-task representation learning: A tensor factorisation approach. arXiv:1605.06391

[39] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *SIGIR*. 729–732.

[40] Yu Zhang and Qiang Yang. 2017. A Survey on Multi-Task Learning. arXiv:1707.08114

[41] Jiejie Zhao, Bowen Du, Leilei Sun, Fuzhen Zhuang, Weifeng Lv, and Hui Xiong. 2019. Multiple Relational Attention Network for Multi-Task Learning. In *KDD*. 1123–1131.

[42] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *RecSys*. 43–51.