Client Time Series Model: a Multi-Target Recommender System based on Temporally-Masked Encoders

Dirk D. Sierag
dirk.sierag@stitchfix.com
Stitch Fix, Inc, Data Science & Machine Learning Department
San Francisco, CA, USA

Kevin Zielnicki
kzielnicki@stitchfix.com
Stitch Fix, Inc, Data Science & Machine Learning Department
San Francisco, CA, USA

ABSTRACT

Stitch Fix, an online personal shopping and styling service, creates a personalized shopping experience to meet any purchase occasion across multiple platforms. For example, a client who wants more one-on-one support in shopping for an outfit or look can request a stylist to curate a 'Fix', an assortment of 5 items; or they can browse their own personalized shop and make direct purchases in our 'Freestyle' experience. We know that personal style changes and evolves over time, so in order to provide the client with the most personalized and dynamic experience across platforms, it is important to recommend items based on our holistic and real-time understanding of their style across all of our platforms. This work introduces the Client Time Series Model (CTSM), a scalable and efficient recommender system based on Temporally-Masked Encoders (TME) that learns one client embedding across all platforms, yet is able to provide distinctive recommendations depending on the platform. An A/B test showed that our model outperformed the baseline model by 5.8% in terms of expected revenue.

CCS CONCEPTS

• **Recommender Systems**; • **Multi-task learning**; • **Neural networks**;

KEYWORDS

industry application, self-attention

ACM Reference Format:
Dirk D. Sierag and Kevin Zielnicki. 2022. Client Time Series Model: a Multi-Target Recommender System based on Temporally-Masked Encoders. In *Sixteenth ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3523227.3547397

1 INTRODUCTION

At Stitch Fix, our goal is to make it easier for people to find the things they love. We do this by creating highly personalized, curated experiences across a range of purchase occasions – whether they want the one-on-one support of a stylist or have a more specific item in mind that they want to discover without getting stuck in a cycle of endless scrolling and filtering. The primary channel that clients begin to interact with our ecosystem is through the Fix. With the Fix, a stylist curates an assortment of five items that reflect a client's style and fit using information they've shared about their fit, size and budget preferences in their profile, as well as any context they've shared in their note requesting a Fix. We send the curated assortment directly to our client, so they can try out the clothes in the comfort of their own home. They keep what they like, and send back anything they don't want to keep along with feedback on why those items didn't work. We recently introduced another curated shopping experience, Freestyle, where our clients can shop directly without the support of a stylist. Clients can discover items or outfits that are curated in their own personal shops. We create these personal shops based on their style profile, style preferences, and past interactions with our platforms. We also have a gamified experience in our ecosystem called Style Shuffle, where our clients can share feedback with us on their personal style in real-time. In Style Shuffle, we'll show a random item to the client and ask them if they would wear the item. The client has the option to give it a thumbs up or thumbs down.

Across all of our shopping experiences, Stitch Fix aims to display items that match the clients style and preferences to make finding what they will like seamless and enjoyable. For Fix, we're providing recommendations to a stylist who is ultimately making the final decision on what to send to the client [2]. So we're showing them the highest matches to their personal style, but the stylist has context of request notes and could push someone out of their comfort zone. For Freestyle, we don't have the stylists curating the final results – we are curating items and ranking them within certain categories or shops based on the client's profile and past interactions. In all cases, the best client experience results from having each platform be informed by interactions on all platforms.

Historically, we have developed specialized recommender systems for each platform, because domain-agnostic models did not perform well. Building and training a separate model for each platform or domain results in increased complexity, especially when signals have to be shared across platforms in real-time [4]. See Figure 1 for an illustration of the model complexity. Moreover, by expanding into different business segments and regions, our codebase and number of models increased significantly. As we grow our business and expand our platforms and client base, the development and maintenance burden of our codebase and models would ever increase. To this extent, we have developed a model that tackles these problems: the Client Time Series Model (CTSM).

RecSys '22, September 18–23, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9278-5/22/09.
https://doi.org/10.1145/3523227.3547397

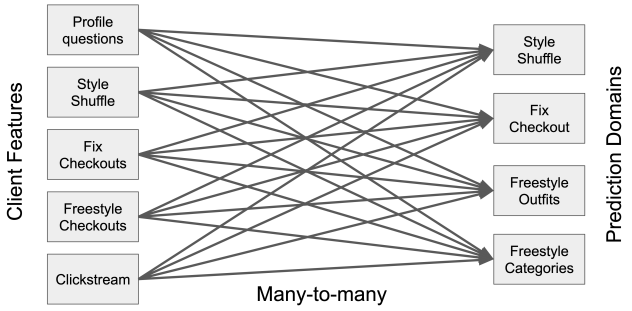

Figure 1: Many-to-many coupling explodes maintenance and complexity.

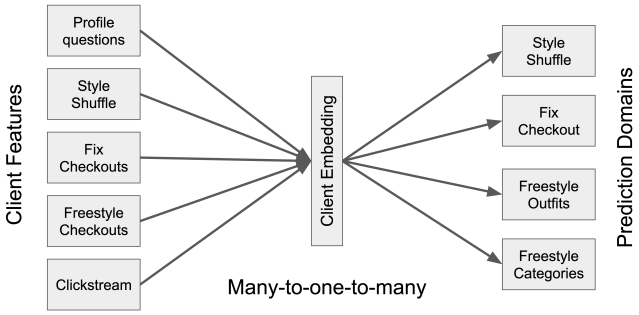

Figure 2: A client embedding can act as a decoupling intermediary.

2 MODEL

The Client Time Series Model estimates one client embedding that serves all domains. This leads to a less complex architecture, compared to a traditional architecture (see Figure 2). In the Client Time Series Model, every client interaction is considered a timestamped event. For example, we do not consider a client's "waist size" to be a static feature. Rather, we consider the event when the client entered their waist size. After all, the client state and preferences are not static, but may evolve over time: in our example, at one moment in time the client waist size might have been 42, but at another moment in time it might change to 41. Using the most recent value of 41 as a feature for interactions that happened at the time it was still 42 would be incorrect. Another example of an event is an interaction with an item: at one moment in time a client might like a certain style of shoes, but as style preferences evolve, the client might prefer a different style of shoes at another moment in time. Both cases demonstrate that a client state evolves over time, and CTSM captures these dynamics.

A consequence is that it is time-safe by design: by considering events chronologically, we avoid the pitfall that past observations are informed by future events, which could typically be a problem when using tabular machine learning models.

Figure 3: Example sequence of client interaction events.

2.1 Temporally-Masked Encoder with Gated Updates

A key component of CTSM is the Temporally-Masked Encoder (TME) with gated updates. The TME is similar to a transformer [3], but is faster and easier to train. Moreover, in our backtesting results TME performed better than a traditional transformer. The main difference is that TME computes weights only once for the whole sequence rather than separately for each input as a function of queries and keys. See Figure 4 for an overview of TME.

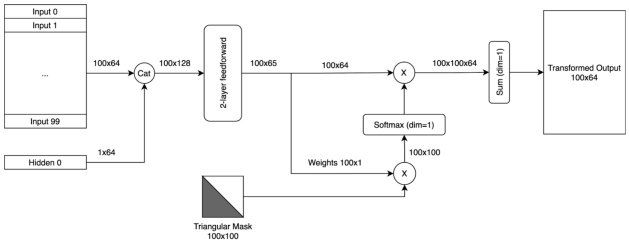

Figure 4: Temporally Masked Encoder (TME).

In the example in Figure 4, the embedding dimension is equal to 64. The input to the masked encoder is a batch of event embeddings for one client, where in the example the batch size is 100. The event embeddings are concatenated with the hidden client embedding, which fans out and is reduced to a smaller dimension using a 2-layer feedforward network. In the example dimension 65: 64 to represent the transformed embeddings and 1 dimension contains the weights to be applied to the mask. This triangular mask masks any events that happen prior to any observation. This can be done so because the events are ordered by timestamp. The final part of the masked encoder returns a weighted masked average.

The output embeddings of the TME are passed through a gated update, which consists of GRUs [1] that are updated in parallel instead of sequentially. This speeds up time and shows empirically proper results. See Figure 5 for an overview.

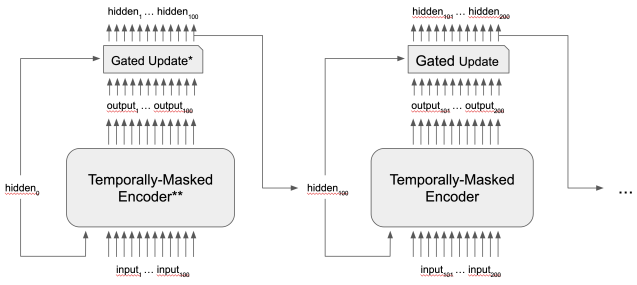

Figure 5: Temporally Masked Encoder wtih Gated Updates.

2.2 Instant reflection of updates to rankings

A key aspect to improve the client experience is to reflect any client's interaction with the platforms instantly in the recommended items. For example, if a client updates their profile, the recommendations on Freestyle should immediately reflect this. Moreover, if a client plays Style Shuffle and subsequently orders a Fix, we want the items that are surfaced to the stylist immediately reflect the client's likes and dislikes. In short, any update on the client profile or client interactions should be instantly reflected in all platforms. CTSM is able to efficiently deal with updates across platforms: At inference time, the only features that have to be passed to the model are all client events that have happened since training, since everything else is reflected in the client embeddings already. Since CTSM doesn't require complex feature engineering or similar, the raw (or almost raw) features can instantly be directed to CTSM as client event updates such that any recommendations reflect those recent events.

2.3 Modeling events

There are two main flavors of events: Updates and Targets.

2.3.1 Updates. Updates are observations of some piece of information that changes our knowledge about the client state. For example, a Fix checkout or a profile change is considered an update. Updates have four components: source, timestamp, client identifier, and payload. The source identifies what type of update we are dealing with. The timestamp identifies when the event took place. The client identifier points to the client to which the update refers. The payload contains any features or information and may differ per source. However, all updates from the same context have the same payload structure. For example, the payload for a Fix checkout might contain an item embedding and an outcome, signifying whether the client purchased the item or not. Each update is then transformed into a fixed dimensional embedding, which is used as an input to the TME. See Figure 6 for an illustration of this process.

2.3.2 Targets. Targets are outcomes of interest to estimate as a function of the client state. An example of a target is the probability that a client purchases an item conditional to the item being sent to the client in a Fix. Each target has an associated loss function, such as binary cross entropy. Targets may correspond to updates, but this is not required. For example, a Fix purchase may be included as an Update as well as a Target. Targets contain the same components as updates, namely source, timestamp, client identifier, and payload, but additionally requires an associated loss function that measures predictions against outcomes from the payload. In our fix checkout example from the updates, the loss function is measured against the 'outcome' inside the payload.

To make a target prediction, the most recent client embedding prior to the target timestamp is fetched, as well as the item embedding. The client vector is then transformed to the item embedding space using a 2-layer feed-forward neural network. The dot product of the two gives us the predicted logits, which can be transformed into the predicted probability by applying the expit function. See Figure 7 for an illustration of this process.

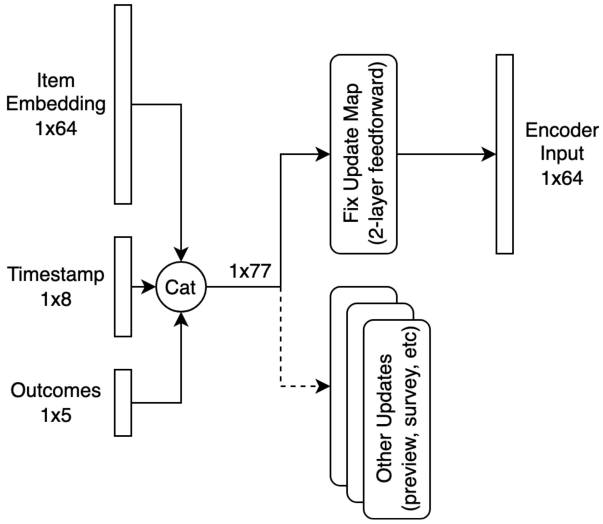

Figure 6: Illustration of processing updates.

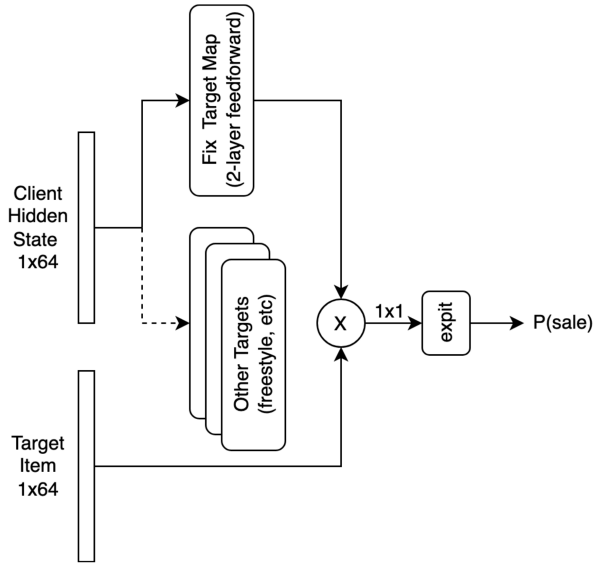

Figure 7: Illustration of target prediction.

3 RESULTS

We have deployed CTSM into our production environment and ran an A/B test in Freestyle. The test showed a 5.8% lift in revenues and a 4.1% lift in order re-engagement, when using CTSM compared to our baseline model.

REFERENCES

[1] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. 103–111.

[2] Eric Colson. 2013. Using Human and Machine Processing in Recommendation Systems. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, Vol. 1. 16–17.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[4] Ruobing Xie, Yalong Wang, Rui Wang, Yuanfu Lu, Yuanhang Zou, Feng Xia, and Leyu Lin. 2022. Long short-term temporal meta-learning in online recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1168–1176.