# Combining Context Features in Sequence-Aware Recommender Systems

**Sarai Mizrachi**
Booking.com
Tel Aviv, Israel
sarai.mizrachi@booking.com

**Pavel Levin**
Booking.com
Tel Aviv, Israel
pavel.levin@booking.com

## ABSTRACT

There are several important design choices that machine learning practitioners need to make when incorporating predictors into RNN-based contextual recommender systems. A great deal of currently reported findings about these decisions focus on the setting where predicted items take on values from the space of sequence items. This work provides an empirical evaluation of some straightforward approaches of dealing with such problems on a real-world large scale prediction problem from the travel domain, where predicted entities do not live in the space of sequence items.

## CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**.

## KEYWORDS

recommender systems; recurrent neural networks; context-aware recommendations; sequence-aware recommendations

## INTRODUCTION

Many types of recommendation problems can be naturally viewed as sequence extension problems. A classic example is a language model offering real-time next word recommendations while typing.

*ACM RecSys 2019 Late-breaking Results, 16th-20th September 2019, Copenhagen, Denmark*

Similarly, when recommending a travel destination we can use the ordered sequence of previously booked destinations as input.

However, some situations require predictions in an output space which is different from the input space. A classic example from the field of natural language processing is document classification: the document is represented by a sequence of words and the prediction happens in the space of possible topics, intents or sentiments. In travel domain we may want to recommend a country to visit next based on user's past history of accommodation bookings (cities, accommodation types, lengths of stay, etc). User history items takes on different values from prediction items.

In both situations (sequence completion and different domain prediction) recurrent neural networks (RNNs) including their gated variants (e.g. GRU [2], LSTM [3]) are commonly used. Both problems become more complex if we have token-level, and/or sequence-level features that we want to factor in. In our destination prediction example we could use a user-level feature such as their home country, as well as booking-specific features (lengths of stay, time since last booking, etc).

This work focuses on the second type of sequence-aware recommendation problem, specifically when we do not assume the predicted items to come from the same space as sequence items. We look at several basic ways of incorporating context into RNN-based recommendation systems and benchmarks their performance.

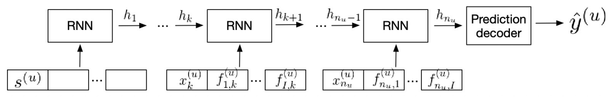

**Figure 1: Sequence-level tokens embedded with items, concatenation**

When we embed items and sequence-level information in the same vector space, we need to "pad" sequence feature embeddings to be of dimension $d_{items} + \sum_1^I d_{f_i}$ since that would be the expected RNN input dimension. The simplest way to achieve this is by introducing extra "dummy" values for each token features whose embeddings would be concatenated to sequence-level feature at the beginning.

## THE SETUP

Our goal is to compare different approaches to account for token- and sequence-level context in RNN-based recommendation systems. An important distinction of this work from much of the previously reported results (e.g. [4], [5], [6]) is that we do not assume that the items in our sequence and the predicted items come from the same space. This set up can be thought of as RNN Encoder $\longrightarrow$ Decoder, where the decoder can be as simple as softmax regression in case of a categorical prediction.

Each user's data point is represented by a sequence of items of $n_u$ items $\mathbf{x}^{(\mathbf{u})} = x_{1:n_u}^{(u)}$, $I$ item-level feature sequences $\mathbf{f}^{(\mathbf{u})} = \{f_{i,1:n_u}^{(u)} : i \in 1, \ldots, I\}$ and sequence-level features $s^{(u)}$. One comparison we do is between two popular techniques for fusing embedded feature information $f_{k,1:n_u}^{(u)}$ with the item embeddings $x_k^{(u)}$ (concatenation vs element-wise multiplication). Another issue we look at is how to best input sequence-level information $s^{(u)}$, by fusing it at each time step with the items along with the token-level features, or by embedding them in the items space and simply using them as additional tokens in the sequence.

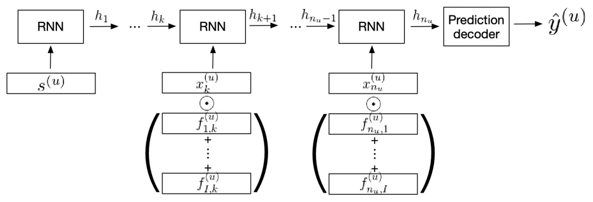

**Figure 2: Sequence-level tokens embedded with items, multiplication**

Unlike in the concatenation example, we don't need to introduce dummy values for token features, or pad embeddings in any way since this approach forces all embeddings to be of the same size and we only perform element-wise operations to merge them.

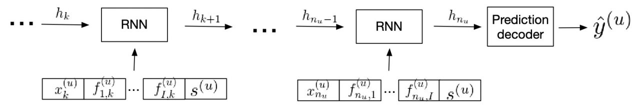

**Figure 3: Sequence-level tokens merged with token-level features, concatenation**

Here we do not use separate sequence tokens to encode sequence-level features. Instead we treat sequence features the same way as token features and concatenate them to the item embeddings.

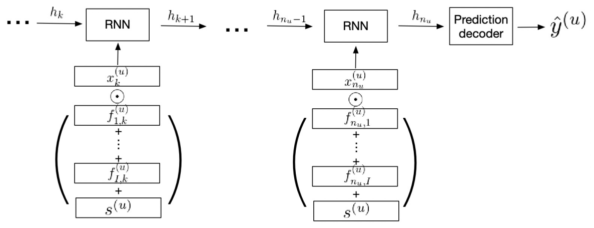

**Figure 4: Sequence-level tokens merged with token-level features, multiplication**

When we merge more than two embeddings through multiplication, we first sum all feature embeddings together and then element-wise multiply the result with item embeddings.

## MODEL ARCHITECTURES

We do our comparisons using four different context-aware RNN-based models, one RNN model with no context and two naive sequence-agnostic baselines.

The RNN architectures use GRU cell as the base and softmax regression to decode sequence representation as a product prediction

$$h_{t+1}^{(u)} = GRU(e_t, h_t)$$
$$\hat{y}^{(u)} = Softmax(Wh_{n_u})$$

where $e_t$ is RNN input for time step $t$ and $W$ is the linear model for multiclass item prediction. The rest of this section will look into how exactly RNN inputs $e_t$ should be derived.

### Baselines

We look at two simple baselines which do not use any sequence, or past history information, and one sequence-aware baseline with no context features. The first baseline is recommending the most popular items based on the last token of the sequence: $\hat{y}^{(u)} = \text{argmax}_y P(y \mid x_{n_u}^{(u)})$. The second baseline is recommending the most popular items according to the sequence-level features: $\hat{y}^{(u)} = \text{argmax}_y P(y \mid \mathbf{s^{(u)}})$. Our third baseline is a standard GRU sequence classification model with no context features.

### Embedding sequence-level features in the items space

The idea behind this approach is that sequence- or user-level features can be simply treated as extra tokens in the sequence. This means that in our RNN architectures those features should be represented in the same space as the items, i.e. we need a single embedding matrix $E \in \mathbb{R}^{(K+M)\times d_{items}}$ to represent $K$ items and $M$ levels of sequence-level features in $d_{items}$ dimensions. All token-level feature would still be embedded in separate vector spaces and represented by matrices $E_j \in \mathbb{R}^{|\mathbf{F_j}|\times d_{F_j}}$, where $d_{F_j}$ is the embedding dimension of feature $j$ and $|\mathbf{F_j}|$ is its cardinality. The following two approaches discuss how we merge token-level embeddings with item embeddings.

*Concatenation merge.* One of the more popular and straightforward approaches for merging item and feature embeddings is simply by concatenating them (see Fig. 1).

$$h_{k+1} = RNN(concat(x_k^{(u)}, f_{1,k}^{(u)}, \ldots, f_{i,n_u}^{(u)}), h_k)$$

One obvious advantage of this approach is the ability to chose different embedding dimensions for each feature $F_i$ according to its cardinality and distributions of values.

*Multiplication merge.* Another popular way of fusing embeddings is through element-wise multiplication (Fig. 2). This approach forces us to have $d_{items} = d_{F_j}, j \in \{1 \dots I\}$. In case when $I > 1$, i.e. we have more than one token-level feature, we follow [1] and first apply element-wise summation to all features, and only then element-wise multiply the result with the item embedding.

$$h_{k+1} = RNN(x_k^{(u)} \odot [f_{1,k}^{(u)} + \dots + f_{i,n_u}^{(u)}], h_k)$$

**Table 1: Precision@k results for baselines and proposed models**

| | Prec@1 | Prec@4 | Prec@10 |
|---|---|---|---|
| *Baselines* | | | |
| Last item only | 0.206 | 0.460 | 0.686 |
| Seq features only | 0.196 | 0.481 | 0.714 |
| Items only (no features) | 0.608 | 0.788 | 0.889 |
| *Seq-level features as seq items* | | | |
| Concatenation | **0.657** | **0.823** | **0.912** |
| Multiplication | 0.648 | 0.811 | 0.904 |
| *Seq-level features as token-level features* | | | |
| Concatenation | 0.656 | 0.822 | 0.911 |
| Multiplication | 0.644 | 0.808 | 0.902 |

### Fusing sequence-level features with the items

Another approach toward sequence-level features that we consider is treating them as additional token-level features. Of course, since sequence-level features do not change across time steps, we merge the same values to each sequence item. As before, we consider two basic merge functions: concatenation and element-wise multiplication.

*Concatenation merge.* The only difference here from the concatenation model above is that now we concatenate an extra feature embedding to our items (Fig. 3). This lets us have shorter sequences, but the input dimension of our RNN needs to be bigger.

*Multiplication merge.* In this scenario (Fig. 4) all embedding dimensions need to be equal. As before, we first sum the feature embedding vectors and then element-wise multiply them with item embeddings.

## DATASET AND EXPERIMENTS

We run our experiments on a proprietary travel dataset of 30 millions travellers from 250 different countries or territories sampled from the last 3.5 years. All users in our dataset made at least three international bookings. To benchmark the performance of our approaches we predict user's last visited country based on a sequence of visited destinations (cities, villages, etc.). The gap between the last booking in the sequence and the target (last visited country) is at least one week.

We used one sequence-level feature, traveler's home country, and two token-level features (days since last trip, accommodation type). Our sequence-level feature clearly takes on different values from our items (countries vs cities), however both are geographical entities, so it is unclear a-priori whether or not embedding them in the same vector space would hurt the performance or not. We evaluate how models perform by measuring precision@k ($k \in \{1, 4, 10\}$) on the data from additional 1,500,000 users. To be consistent we use embedding dimension of 50 for all items and features in all models. The GRUs have two layers and 500 hidden units. Our main sequence tokens (destinations) have cardinality of 50,000. The sequence-level home country feature is of size 250. For token-level features there are 1,244 possible values for "days since last booking" and 31 different accommodation types . The target is one of the 250 possible countries (or territories).

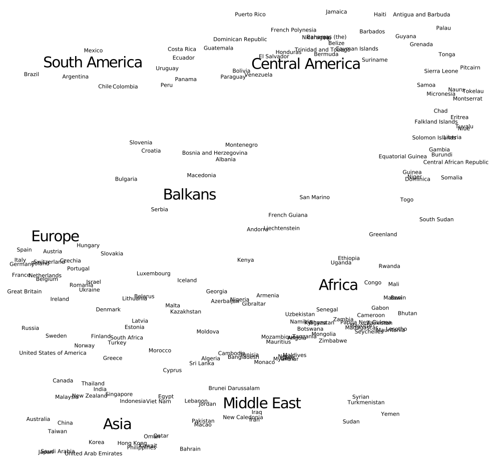

**Figure 5: Country embeddings**

Country embeddings as sequence-level features from model in Fig 4 visualized 2D. The model successfully learned reasonable clustering of countries without any direct supervision related to their locations.

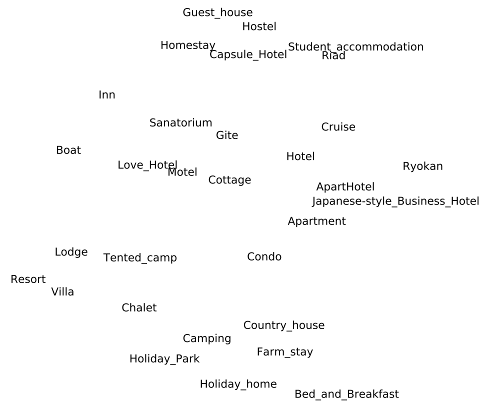

**Figure 6: Accommodation types**

Visualization of a token-level feature "accommodation type" from the model in Fig. 4. Similar property types tend to be closer together in the embedding space

Table 1 shows the precision@k results for the models. Concatenation seems to perform better than multiplication for both ways of inputting sequence-level features. All sequence recommenders do significantly better than our naive baselines. Our featureless sequence recommender baseline also significantly outperforms the naive baselines, but noticeably worse than the context-aware models. On the other hand, the choice of inputting sequence-level features as items, although slightly better, seems to matter much less in terms of model accuracy.

## DISCUSSION

We have looked at simple ways of merging item and feature embeddings in RNN-based recommendation problems where sequence items and prediction items take on values from different spaces. Our main conclusion is that for simple RNN-based sequence models concatenating features seems to work better than merging them element-wise, while our choice of how we input our sequence-level features (as extra items or as token features) matters less. Despite the limited scope of our study, we believe it will help to guide machine learning practitioners in designing more effective architectures that are able to incorporate both sequence- and item-level context into RNN-based recommender systems. We have only analyzed the case of a single sequence-level feature of relatively small cardinality. In follow-up work it would be beneficial to look at more general cases of multiple sequence-level features and various strategies to fuse them together, along with item-level information. It is also important to look at more complex merge functions, such as feedforward neural networks or bilinear forms in future research.

## REFERENCES

[1] 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 *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*. ACM, New York, NY, USA, 46–54. https://doi.org/10.1145/3159652.3159727

[2] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1724–1734. https://doi.org/10.3115/v1/D14-1179

[3] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[4] Q. Liu, S. Wu, D. Wang, Z. Li, and L. Wang. 2016. Context-Aware Sequential Recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 1053–1058. https://doi.org/10.1109/ICDM.2016.0135

[5] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *ACM Comput. Surv.* 51, 4, Article 66 (July 2018), 36 pages. https://doi.org/10.1145/3190616

[6] Elena Smirnova and Flavian Vasile. 2017. Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks. In *Proceedings of the 2Nd Workshop on Deep Learning for Recommender Systems (DLRS 2017)*. ACM, New York, NY, USA, 2–9. https://doi.org/10.1145/3125486.3125488