Hotel2vec: Learning Hotel Embeddings from User Click Sessions with Side Information

IOANNIS PARTALAS, Expedia Group, Switzerland
ANNE MORVAN, Expedia Group, Switzerland
ALI SADEGHIAN*, University of Florida, USA
SHERVIN MINAEE*, Snap, Inc, USA
XINXIN LI*, Expedia Group, USA
BROOKE COWAN*, Apex Clearing Corporation, USA
DAISY ZHE WANG, University of Florida, USA

We propose a new neural network architecture for learning vector representations of items with attributes, specifically hotels. Unlike previous works, which typically only rely on modeling of user-item interactions for learning item embeddings, we propose a framework that combines several sources of data, including user clicks, hotel attributes (e.g., property type, star rating, average user rating), amenity information (e.g., if the hotel has free Wi-Fi or free breakfast), and geographic information that leverages an hexagonal geospatial system as well as spatial encoders. During model training, a joint embedding is learned from all of the above information. We show that including structured attributes about hotels enables us to make better predictions in a downstream task than when we rely exclusively on click data. We train our embedding model on more than 60 million user click sessions from a leading online travel platform, and learn embeddings for more than one million hotels. Our final learned embeddings integrate distinct sub-embeddings for user clicks, hotel attributes, and geographic information, providing a representation that can be used flexibly depending on the application. An important advantage of the proposed neural model is that it addresses the cold-start problem for hotels with insufficient historical click information by incorporating additional hotel attributes, which are available for all hotels. We show through the results of an online A/B test that our model generates high-quality representations that boost the performance of a hotel recommendation system on a large online travel platform.

CCS Concepts: • **Computing methodologies** → **Ranking**; **Learning latent representations**; **Learning from implicit feedback**; **Neural networks**.

Additional Key Words and Phrases: neural networks, embeddings

1 INTRODUCTION

Learning semantic representations of different entities, such as textual, commercial, and physical, has been a recent and active area of research. Such representations can facilitate applications that rely on a notion of similarity, for example recommendation systems and ranking algorithms in e-commerce [2, 5, 6, 9, 18, 22, 39].

In natural language processing, word2vec [28] learns vector representations of words from large quantities of text, where each word is mapped to a d-dimensional vector such that semantically similar words have geometrically closer vectors. This is achieved by predicting either the context words appearing in a window around a given target word (skip-gram model), or the target word given the context (CBOW model). The main assumption is that words appearing frequently in similar contexts share statistical properties (the *distributional hypothesis*). Crucially, word2vec models, like many other word embedding models, preserve sequential information encoded in text so as to leverage word co-occurrence statistics. The skip-gram model has been adapted to other domains in order to learn dense representations

*Work completed while author was at Expedia Group.

of items other than words. For example, product embeddings in e-commerce [12] or vacation rental embeddings in the hospitality domain [11] can be learned by treating purchase histories or user click sequences as sentences, and applying a word2vec approach.

Most of the prior work on item embedding exploit the co-occurrence of items in a sequence as the main signal for learning the representation. One disadvantage of this approach is that it fails to incorporate rich structured information associated with the embedded items. For example, in the travel domain, where we seek to embed hotels and other travel-related entities, it could be helpful to encode explicit information such as user ratings, star ratings, hotel amenities, and location in addition to implicit information encoded in the click-stream.

In this work, we propose an algorithm for learning hotel embeddings that combines sequential user click information in a skip-gram approach with additional structured information about hotels. We propose a neural architecture that adopts and extends the skip-gram model to accommodate arbitrary relevant information of embedded items, including but not limited to geographic information, ratings, and item attributes. In experimental results, we show that enhancing the neural network to jointly encode click and supplemental structured information, outperforms a skip-gram model that encodes the click information alone. The proposed architecture also naturally handles the cold-start problem for hotels with little or no historical clicks. Specifically, we can infer an embedding for these properties by leveraging their supplemental structured metadata.

Compared to previous work on item embeddings, the novel contributions of this paper are as follows:

(1) We propose a novel yet straightforward framework for fusing multiple sources of information about an item (such as user click sequences and item-specific information) to learn item embeddings via self-supervised learning.
(2) We generate an embedding that consists of three sub-embeddings for clicks, geography, and amenities attributes, which can be employed either as separate component embeddings or a single, unified embedding.
(3) We address the cold-start problem by including hotel metadata which are independent of user click-stream interactions and available for all hotels. This helps us to better impute embeddings for sparse items/hotels.
(4) We show significant gains over previous work based on click-embeddings in several experimental studies.

The structure of the remainder of this paper is as follows. Section 2 gives an overview of some of the recent works on neural embedding. Section 3 provides details of the proposed framework, including the neural network architecture, training methodology, and how the cold-start problem is addressed. In Section 4, we present experimental results on several different tasks and a comparison with previous state-of-the-art work. Section 5 highlights online A/B tests obtained for ranking hotels on a search result page by including as features these embeddings in the search ranking model. Section 6 concludes the paper.

2 RELATED WORK

2.1 Embeddings from user sequences for different application domains

Recommendation is an inherently challenging task that requires learning user interests and behavior. There has been a significant body of research on advancing it using various frameworks [3, 13, 25, 30, 42]. Learning a semantic representation/embedding of the items being recommended is a critical piece of most of these frameworks. Building recommender systems for hotels is an specially hard task due to challenges such as balancing between the popular hotels and the newly added ones (without enough clicks), and the very large space of candidates.

Neural network models have been widely used for learning embeddings from user sessions [37, 38]. One prominent use case is learning product embeddings for e-commerce. In [4, 12], the authors develop an approach based on the skip-gram model [28], frequently used in natural language processing. They leverage users' purchase histories obtained from their e-mail receipts to learn a dense representation of products. Each user's complete purchase history is represented as a sequence, which is treated as a sentence in which the items are considered words. For music recommendation, authors in [16] define track representations considered as ground truth with a word2vec continuous bag-of-word model and a music session (a collection of tracks) as the average of track embeddings it contains. A session-level user embedding is further learned via a LSTM model to maximize the cosine similarity between the predicted user session-level embedding with the observed ground truth music session representation. In the online travel space, authors in [11] use the skip-gram framework to learn embeddings for vacation rental properties. They extend the ideas in [12] to take into account a user's click stream data during a session. A key contribution of their method is the modification of the skip-gram model to always include the booked hotels in the context of each target token, so that special attention is paid to bookings. They also improve negative sampling by sampling from the same market, which leads to better within-market listing similarities. Nevertheless, their model relies exclusively on large amounts of historical user engagement data, which is a major drawback when such data are sparse.

2.2 Link to graph approaches

The skip-gram loss can be seen also as a graph-based one and this paves the way to graph embedding approaches which share the same similarity assumption. Graph embedding methods aim at learning in an unsupervised manner embeddings for pairs of edge-linked nodes which are more similar to each other than the embeddings of pairs of nodes without an edge between them. In that case, the considered graph is constructed from the co-clicks: nodes are the items and an edge designates a co-click between two items. In this area, graphSAGE [15] is the first approach to work in the inductive mode. By learning aggregation functions, graphSAGE predicts the embedding of a *new* node without a re-training based on its features and neighborhood. Two recent methods PyTorch-BigGraph (PBG) [24] and Cleora [33] offer more scalability by partitioning the graph. PBG proposes a margin-based ranking objective function between positive and negative pairs of nodes. Cleora relies exclusively on the graph structure and does not use a contrastive learning objective, hence does not require to sample positive or negative examples. Instead Cleora obtains node embeddings by iteratively aggregating each node's neighbor embeddings followed by an L_2-normalization. Cleora prevents the embeddings from collapsing through a careful initialization and the normalization step.

2.3 Merging all side information to capture "context" of user activity

In another relevant work, [7], authors propose a framework for YouTube video recommendation which fuses multiple features (e.g., video watches, search tokens, geo embeddings) into a unified representation via a neural architecture. They then use these embeddings for candidate generation and ranking. The main limitation of this work is that the individual embeddings are learned separately, and then combined via a neural network to perform classification.

There are also several works which try to use attention mechanism as a tool to capture "context" of users' activities on the basis of actions they have performed recently, such as contextual self-attention network for user sequential recommendation [19], multi-pointer co-attention network [35], multi-order attentive ranking model [40], neural attentive interpretable recommendation system [41], self-attentive sequential recommendation [21, 29].

Similar to our work on hotel2vec, there are also some works which attempt to include explicit item attributes (e.g., size, artist, model, color) within the sequence prediction framework using various strategies. In [36], the item metadata

is injected into the model as side information to regularize the item embeddings. Their approach only uses one feature (singer id) in the experiments. In addition, it does not accommodate learning independent embedding vectors for each attribute group. Most recently, [34] propose a method where they train separate encoders for text data, click-stream session data, and product image data, and then use a simple weighted average to unify these embeddings. The weights are learned using grid search on the downstream task. While their approach allows for exploring independent embedding vectors, the sub-embeddings of different attribute groups are learned independently rather than jointly. In addition to efforts extending the skip-gram framework, emerging research attempts to extend GloVe [31] by incorporating various attributes. Authors in [20] incorporate attribute information into GloVe by modifying the loss function such that the representation of a location can be learned by combining both text and structural data.

3 THE PROPOSED FRAMEWORK

Similar to word2vec [28], by treating the clicks made by users within an interactive web session as words, and sequences of clicks as sentences, we seek to predict the context hotels (words), given a target hotel (word) in the session (sentence). On a high level, this is the approach proposed in [11, 12]. We refer to this approach as a *session-only* model.

As mentioned earlier, one drawback of this approach is that it does not use any information apart from the click data, making it very challenging to make predictions for unseen hotels or hotels with sparse click data. In addition, the model may be forced to learn certain semantic features which capture aspects of user interest, hotel geographic information, hotel attributes, and so on, as latent variables as opposed to leveraging them as explicitly-provided input features. To address these shortcomings, we propose adding more explicit information about the hotel as model input. Intuitively, this should make the model more efficient during training as well as provide information that it can use when making predictions on unseen or sparse hotels.

Another major advantage of our model is its use of different projection layers for various hotel/item attributes. This enables us to learn independent embedding vectors representing different facets of the property, in addition to an enriched, unified embedding for each hotel. This model also provides a dynamic framework for updating the embedding of a hotel, once its user-rating or other attribute information changes over time. This is not trivial in session-only models, unless we re-train a new model based on recent click data post attribute changes. In the remainder of the paper, we refer to our proposed hotel2vec model as an *enriched* model, in contrast to the *session-only* model introduced above.

3.1 Neural Network Architecture for the enriched hotel2vec

Figure 1 illustrates the proposed architecture for an *enriched* hotel2vec model. Each aspect of the hotel 1) the one-hot encoding of the clicked property ids within the same session, 2) the associated amenities and 3) the geographical information is embedded separately, and these representations are later concatenated and further compressed before being used for context prediction. Formally, a click session is defined as a sequence of hotels (items) $\{h_1, h_2, \cdots, h_n\}$ clicked on by a user during a defined window of time or visit. For a given hotel, we denote the click, amenity, geographic, and enriched embedding vectors with $\mathbf{V}_c$, $\mathbf{V}_a$, $\mathbf{V}_g$, and $\mathbf{V}_e$ respectively. These are defined as follows:

$$\begin{aligned} \mathbf{V}_c &= f(I_c; \mathbf{W}_c) \\ \mathbf{V}_a &= f(I_a; \mathbf{W}_a) \\ \mathbf{V}_g &= f(I_g; \mathbf{W}_g) \\ \mathbf{V}_e &= \text{ReLU}([\mathbf{V}_c, \mathbf{V}_a, \mathbf{V}_g]^\top \mathbf{W}_e) \end{aligned} \tag{1}$$

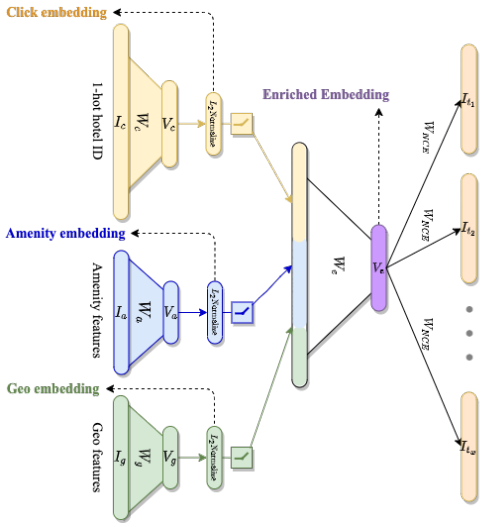


Fig. 1. The 3-block-diagram of the *enriched* hotel2vec model with a single encoding layer.

where I_c is the one-hot encoding of hotels in the click session, and $I_g \in \mathbb{R}^g$ is a continuous vector with geographical information of the hotel which we will explain later what type of information contains. $f(x; \mathbf{W})$ is a normalized projection layer parameterized with trainable weights $\mathbf{W}$, i.e., $f(x; \mathbf{W}) = \text{ReLU}(\frac{x\mathbf{W}}{\|x\mathbf{W}\|_2})$ where ReLU is the rectified Linear Unit activation function [10].

3.1.1 Geographical features. As geographical features we use a spatial encoder [26] (later called space2vec) as well as the H3 hierarchical geo-spatial system[1]. H3 maps the world in hexagons which can be defined in different resolutions. Both space2vec and H3 embeddings are concatenated to form I_g.

To obtain space2vec embedding, we encode a spatial point x (PE for Point Encoder) as a concatenation of multi-scale representations where S is the total number of grid scales

$$[PE_0^k(x); \ldots; PE_s^k(x); \ldots; PE_{S-1}^k(x)]$$

and each component of x is used separately (in our case the latitude and longitude of the hotel) as follows:

$$\forall s \in \{0, \ldots, S-1\},\ PE_s^k(x) = [PE_{s,1}^k(x); PE_{s,2}^k(x)]$$

$$\forall l \in \{1, 2\},\ PE_{s,l}^k(x) = [\cos \frac{x[l]}{\lambda_{min} \cdot k^{s/(S-1)}}; \sin \frac{x[l]}{\lambda_{min} \cdot k^{s/(S-1)}}]$$

where λ_{min} and λ_{max} are the minimum and maximum grid scale and $k = \frac{\lambda_{max}}{\lambda_{min}} = \frac{10}{1000}$. The multi-scale representation is encoded with a fully connected layer in the hotel2vec model. As activation function we tried both ReLU and linear one and we found out that linear works the best.

To obtain H3 embedding, we use the index at resolution 8 which we find reasonable for our use case. From the latitude and longitude of a hotel we can get the unique id (given a resolution) of the H3 hexagon which we embed in the model.

[1] https://eng.uber.com/h3/

3.1.2 Amenity features. Amenity features (e.g., PetsAllowed, GuestRating, SpaServices, etc.) can be categorical or numerical with possible missing values. Thus, $I_a \in \mathbb{R}^{58}$ is partitioned per feature, where for numerical features we simply use an element of I_a assigned with the value of that feature, and for categorical features with m categories, we assign m elements of I_a and set the corresponding category to 1 and the others to 0. If the feature is missing, we set everything to 0.

3.1.3 Loss function. We train our model by optimizing the Noise Contrastive Estimation (NCE) loss [14]. More formally, given h_t as the target, we estimate the probability of h_c being a context hotel to be

$$\log P(h_c|h_t) = \log \sigma(\mathbf{V}_{e_t}^{\top} \mathbf{W}_{c,:}) \tag{2}$$

where σ is the sigmoid function, $\mathbf{V}_{e_t}$ is the enriched embedding of h_t, $\mathbf{W}_{c,:}$ is the c^{th} row of the output projection weights, W_{NCE}. The $I_{t_1}, \cdots, I_{t_\omega}$ vectors in Figure 1 represent the one-hot encodings of other hotels in the training window of I_c (in the window, stride length is 2 before and after the hotel). We find parameters of the model by maximizing the probabilities of correct predictions. We train the model using backpropagation and by minimizing the following loss function (L_2-regularization is also applied):

$$\mathcal{J}(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \left(\log P(h_{c_t}|h_t) + \sum_{h_i \in \mathcal{N}_c} \log \sigma(-\mathbf{V}_{e_t}^{\top} \mathbf{W}_{h_i,:}) \right) \tag{3}$$

where θ includes all the parameters of the model, T is the size of the batch, $\mathbf{W}_{i,:}$ is i^{th} row of W_{NCE}, $\mathcal{N}_c = \{h_i | 1 \leq i \leq N, h_i \sim P_n(h_c)\}$ is the set of negative examples, and $P_n(h_c)$ is the distribution which we use to pick the negative samples which we discuss in Section 3.2. Finally, we train the model by maximizing Eq. 3 using batch stochastic gradient descent.

3.2 Negative Sampling

It is well known [14, 28] that using negative sampling, a version of noise contrastive estimation, significantly decreases the amount of time required to train a classifier with a large number of possible classes. In the case of recommendation, there is typically a large inventory of items available to recommend to the user, and thus we train our skip-gram model using negative sampling. However, it is not uncommon that users frequently search exclusively within a particular subdomain. For example, in hotel search, a customer looking to stay in Miami will focus on that market and rarely across different markets. This motivates a more targeted strategy when selecting negative samples: we select negative samples within the market that the target and context hotels belong to. Throughout this paper, a *market* is defined as a set of hotels in the same geographic region. It's worth noting that there may be multiple markets in the same city or other geographical region.

3.3 Cold Start Problem

In practice, many hotels/items appear infrequently or never in historical data. Recommender systems typically have difficulty handling these items effectively due to the lack of relevant training data. Apart from the obvious negative impacts on searchability and sales, neglecting these items can introduce a Matthew's effect where "rich get richer and the poor get poorer". That is, the less these items are recommended, or the more they are recommended in inappropriate circumstances, the more the data reinforces their apparent lack of popularity.

Dealing with such hotels/items and choosing appropriate weights for them is referred to as the "cold start problem." One of the main advantages of the enriched hotel2vec model over session-only approaches is its ability to better handle cold start cases. Although an item might lack sufficient prior user engagement, there are often other attributes available. For example, in our use case, thousands of new properties are added to the lodging platform's inventory each quarter. While we do not have prior user engagement data from which to learn a click embedding V_c, we do have other attributes such as geographical location, star rating, amenities, etc. hotel2vec can take advantage of this supplemental information to provide a better cold-start embedding. For newly listed hotels with no click-session information, one can simply choose V_c for new hotels at random and hotel2vec computes V_e using the randomly initialized V_c and the other hotel attributes which are known even for recently listed hotels. In Section 4.4, we compare with the *session-only* model [11] when setting V_c as the average of other hotels' embeddings in the same market and show a 70% gain on Hits@10 for the cold start hotels.

4 EXPERIMENTAL RESULTS

In this section, we present several experiments to evaluate the performance of the trained hotel2vec embeddings. We refer the reader to next Section 5 for results on an online A/B testing. Before diving into the details of the experiments, we first describe the dataset and model parameters.

4.1 Experimental Framework

4.1.1 Real-world large-scale dataset from Expedia Group, a leading Online Travel Agency (OTA). Our dataset collected in 2019 (pre-Covid period) contains more than 65 million user click sessions, which includes more than 1.4 million unique hotels. A click session is defined as a span of clicks performed by a user with no gap of more than 7 days for the same destination and search parameters. Data are summarized in Table 1. We randomly split the sessions into training, validation, and test with a ratio of 8:1:1.

Table 1. Dataset statistics.

Number of user click sessions	65M
Number of unique hotels	1.4M
Avg # of clicks per user session	6
Min. # of clicks per user session	2
Max. # of clicks per user session	50

4.1.2 Experiment configuration. We use a system with 64GB RAM, 8 CPU cores, and a Tesla V100 GPU. We use Python 3 as the programming language and the Tensorflow [1] library for the neural network architecture and gradient calculations. All weight matrices are initialized with a he_normal [17] initializer, and click embedding vectors are initialized uniformly at random. As mentioned previously, L_2-regularization is applied on the weights which we add in the loss function.

4.1.3 Downstream tasks. In the following sections, we provide the performance evaluation of our trained embeddings on several experimental tasks. We start with the quantitative results focusing on the next-item prediction task based on model's output probabilities (Section 4.2.1) and cosine similarity (Section 4.2.2), then present some qualitative results.

We also evaluate each model's performance on the cold start problem and provide insights on the effect of some of the hyper-parameters.

4.1.4 Comparison against state-of-the-art embedding baselines. For next-item prediction task based on cosine similarity in Section 4.2.2, we compare results against the following embedding methods:

- the state-of-the-art *session-only* model proposed in [11]. As explained in Section 2, this model can only learn from historical user click sessions without direct use of the item's attributes.
- our hotel2vec which combines both hotel/item attributes and the click session data.
- Matrix Factorization (MF) approach [23], where we factorize the matrix of the co-occurrence of the clicked hotels. Specifically, we factorize the log of the co-occurrence matrix.
- Cleora [33] and,
- graphSAGE [15] presented in Section 2. We used the implementation from Stellargraph [2].

For all considered experiments, we tune the hyperparameters of all models on the validation set. In particular for the state-of-the-art baseline session-only model [11], we search for a learning rate from $\{0.01, 0.1, 0.5, 1.0, 2.5\}$ and embedding dimensions from $\{32, 128\}$. To train the model weights, we use stochastic gradient descent (SGD) with exponential decay (power=0.99 and staircase steps per 40k training steps) since it performed better than other optimizers in our case, and a batch size of 1024. We found that a learning rate of 0.5 and an embedding dimension of 32 worked best. Hence, throughout the remainder of the paper, all embeddings will have dimension 32.

For hotel2vec, an initial learning rate of 0.05 worked best; for the dimensions of the embedding vectors, we found that letting V_c, $V_e \in \mathbb{R}^{32}$, $V_a \in \mathbb{R}^{15}$ and $V_g \in \mathbb{R}^{36}$ worked best. For the multi-scale parameter in space2vec module we tuned S to 16 and the output dimension to 28 with linear activation. For H3 layer we set the embedding size to 8 also. These two representations are concatenated to form the geo-embedding part of the model. For both session-only [11] and hotel2vec models, the number of negative samples is 2000.

For the MF approach we constructed the co-click matrix using a skip window of size 2 and factorized it with Alternate Least Squares algorithm where we tuned the maximum iterations at 10 and the regularization parameter at 0.02. Note that as the result are two matrices, one for the target hotel and one for the context hotel, we obtained the best results by averaging the two corresponding vectors for each hotel in order to obtain the final representation.

To make fair comparison with graphSAGE, we choose then 200 neighbors to sample from the direct neighborhood and 10 from the 2-hop neighborhood. Stellargraph's implementation of graphSAGE samples as many negative as positive neighbors. We then follow recommendations of the authors and set no dropout, L_2-regularization and ADAM for the optimizer. The binary cross entropy loss is used for the link prediction task to learn the embeddings as well as the sigmoid function for the activation.

4.2 Quantitative Analysis: Next-item prediction task

A robust metric for evaluating a set of hotel embeddings (or, more generally, any set of items displayed to a user in response to an information need) is its ability to predict a user's next click/selection. In this section, we compare our model based on the Hits@k and MRR@k metrics in various cases.

[2] https://github.com/stellargraph/stellargraph

- Hits@k measures the average number of times the correct selection (i.e. the hotels clicked by the users in a session) appears in the top k predicted hotels (i.e. the hotels with highest predicted probabilities, conditioned on the current hotel).
- MRR@k (for Mean Reciprocal Rank) evaluates the average list quality of k items returned by the model (ordered by predicted probabilities) by looking at the rank of the first correctly predicted item.

4.2.1 Next-item prediction task based on model's output probabilities. We consider two main scenarios:

- *Raw* evaluation: We are given the current hotel clicked by the user, and we try to predict the next clicked hotel among all approximately 1.4M hotels.
- *Filtered* evaluation: The second scenario is identical except we limit the candidates to hotels within the same market.

For the last scenario, three simple baselines are also included where we rank the properties according to their average guest rating and their last week and last year popularity which is the raw number of bookings that the property received the last seven days and last twelve months respectively.

Table 2 shows Hits@k and MRR@k for $k \in \{10, 100\}$ for hotel2vec and Session-32 [11] approach. We also present results for hotel2vec when we drop the geographical features.

We notice that hotel2vec outperforms the session-only model by a huge margin, demonstrating the utility of including item attributes when learning embeddings. By removing the geographical part we experience a drop in all the metrics for hotel2vec.

We also compare both models in the filtered scenario. This is a more realistic case because limiting hotels to the same market reduces the effect of other information the recommender system can use to provide more relevant suggestions to the user. Table 2b shows predictions results in the filtered scenario. The proposed hotel2vec model outperforms the "highest rated" and "most popular" baselines with a large margin.

Table 2. Prediction results of the most likely hotel the user will click next among all possible hotels.

(a) Results for the *raw* evaluation (no restriction on the candidates).

Methods	Hits@10	Hits@100	MRR@10	MRR@100
Session-32 [11]	0.1565	0.5352	0.0512	0.0689
hotel2vec-no-geo	0.1763	0.5585	0.0587	0.0728
hotel2vec	**0.1807**	**0.5671**	**0.0604**	**0.0746**

(b) Results for the *filtered* evaluation, when hotel candidates are restricted to the same market as the current hotel.

Methods	Hits@10	Hits@100	MRR@10	MRR@100
Highest Rated	0.0158	0.0102	0.0029	0.0032
Most Popular (last year)	0.0739	0.1789	0.0187	0.0230
Most Popular (last week)	0.0928	0.2397	0.0233	0.0292
Session-32 [11]	0.1583	0.5562	0.0605	0.0752
hotel2vec	**0.1998**	**0.5987**	**0.0675**	**0.0825**

As demonstrated by Table 2, the hotel2vec model outperforms the baseline session model from [11] significantly in both scenarios. This shows the effectiveness of hotel2vec in incorporating both click sessions and item/hotel attributes for better recommendations.

4.2.2 Next-item prediction task based on cosine similarity. In this section, rather than using the model's output probabilities to induce a ranking over hotels, we measure Hits@k and MRR@k over the ranking induced using cosine similarity of the embedding vectors. This is useful in scenarios where it is not feasible to directly use the model's probabilities. In particular, it is easier to compare the different baselines based solely on the embeddings, that is why we present here more competing baselines. Table 3 shows the results for various embeddings: Session-32 [11] and hotel2vec but also Cleora [33], MF [23] and graphSAGE [15]. The simple baselines "Highest rated", "Most Popular (last year)", "Most Popular (last week)" which are not embedding approaches were removed since they were already performing poorly in the previous experiment. For conciseness, we focus only on the *raw* evaluation scenario.

Hotel2vec embeddings achieve the highest performance. We believe that bad performance obtained by graphSAGE is due to its scalability issue. By increasing the number of sampled neighbors, it would probably increase the metrics'results but training time would be prohibitive (about 150h of training were already needed to obtain these results). We also see from Table 3 that using cosine similarity instead of the whole network does not result in a huge decrease in performance.

Table 3. Results of predicting the next click among all possible hotels using cosine similarity of the vectors (*raw* evaluation scenario).

Vector used in cosine similarity	Hits@10	Hits@100	MRR@10	MRR@100
graphSAGE [15]	0.0040	0.0212	0.0009	0.0014
MF [23]	0.964	0.4689	0.0297	0.0417
Cleora [33]	0.1360	0.4160	0.0350	0.0449
Session-32 [11]	0.141	0.491	0.0389	0.0512
hotel2vec	**0.1676**	**0.5341**	**0.0413**	**0.0546**

4.3 Qualitative Analysis

The learned hotel embeddings can be used for recommending similar hotels in various situations. In this section, we show examples of how these embeddings are helpful with real examples of hotels from our dataset.

4.3.1 Visualization of embedding clusters. To further illuminate the nature of the embeddings learned by the hotel2vec model, we examine a low-dimensional projection (UMAP [27]) of hotel embeddings in the Miami market (Fig. 2b and 2a). The colors signify the grouping of hotels into various competing subcategories (i.e., similar hotels), manually annotated by a human domain expert. The hotel2vec model is significantly better at clustering similar hotels than the session-only model [11].

4.3.2 Finding the top-k most similar hotels. A common scenario is finding similar hotels to a target hotel in other destinations. For example, when the user searches for a specific hotel name (e.g., Hotel Beacon, NY) we would like to be able to recommend a few similar hotels. The learned embeddings can be used to find top-k most similar hotels to a given one. Given a target hotel h, we compute the cosine similarity of every other hotels with h and pick the most similar hotels. Rigid evaluation of this system requires A/B testing; here we show a few examples comparing our hotel2vec embeddings and the session-only [11] embeddings in Fig. 3 to provide some intuition for the behavior of the two models.

(a) hotel2vec embeddings (b) Session-32 [11] embeddings

Fig. 2. Low-dimensional visualization (UMAP [27]) of hotel embeddings from the Miami area. Different colors represent expert annotations of competing hotels. Our model has successfully captured most of the similarities.

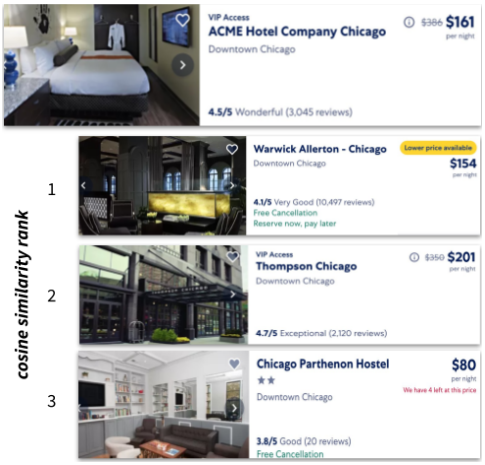


Fig. 3. Example of recommendations based on cosine similarity of hotel2vec embedding vectors. Ranking by the session-32 [11] model placed 3rd before 1st (3,1,2), though it is a hostel, cheaper, and has a lower user rating than the target hotel.

4.4 Addressing the Cold Start Problem

Here we analyze how well the model learns embeddings for hotels with no presence in the training data. To demonstrate the effectiveness of our model, we compare the hotel2vec's Hits@k with the session-only model [11]'s Hits@k, for target hotels that were absent during training for same-market predictions (*filtered* evaluation). For hotel2vec, cold-start concerns the V_c embedding associated to co-clicked properties within the same session. We use a simple heuristic for cold-start imputation and compare the results with hotel2vec model for cold-start hotels. To impute vectors for cold-start hotels, we borrow the idea in [11] and use price, star rating, geodesic distance, type of the property (e.g., hotel, vacation rental, etc.) size in terms of number of rooms, and the geographic market information. For each imputed property, we collect the top 100 most similar properties in the same market based on the above features, considering

only those properties that fall within a radius of 5km of the target hotel and for which we have an existing V_c (resp. session-32) embedding. We then average these embeddings to obtain the V_c (resp. final) embedding of the hotel to impute. Results are shown in Table 4. The proposed enriched embedding by hotel2vec model significantly outperforms the session-based embeddings for cold-start hotels.

Table 4. Cold start experiments: Same-market prediction results when the target hotel is an unseen hotel, click embeddings imputed by averaging the top-100 similar hotel embeddings in market.

Imputed	Hits@10	Hits@100	MRR@10	MRR@100
Cleora [33]	0.0131	0.0067	0.0019	0.0022
MF [23]	0.0196	0.0380	0.0053	0.0061
Session-32 [11]	0.0296	0.0632	0.0079	0.0093
hotel2vec	**0.0513**	**0.1248**	**0.0132**	**0.0162**

4.5 Training Convergence Analysis

In this section, we first look at the learning curves for both the session-32 [11] and hotel2vec models. Then, we analyse the effect of N (number of negative samples), and lr (learning rate) alongside the optimization algorithm on the performance of our model.

4.5.1 Learning curves. Fig. 4 shows the overall training progress of both the session-32 [11] and hotel2vec models with their respective best hyperparameters optimized for Hits@100. Our model achieves similar performance with fewer data.

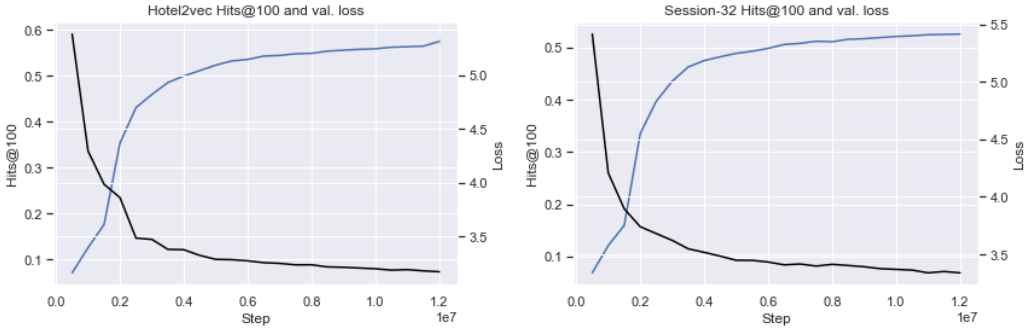
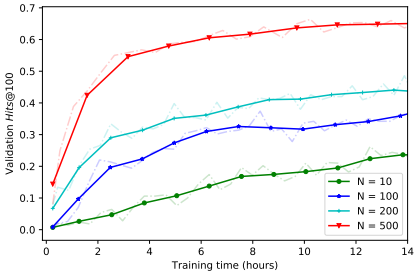


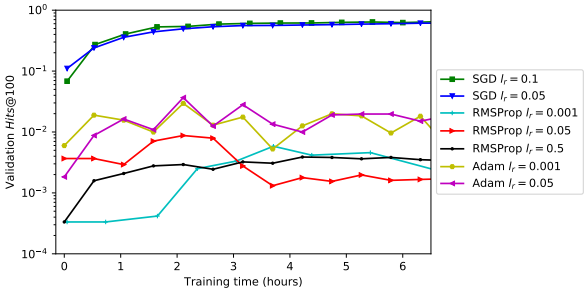
Fig. 4. Training progress of both models the session-32 [11] and hotel2vec models with their respective best hyperparameters optimized for negative sampling loss on validation set and Hits@100.

4.5.2 Number of negative samples. An interesting phenomenon is the effect of increasing the number of negative samples on training time and accuracy. Although it takes more time to create a large number of negative samples, as Fig. 5a shows, using more negative samples results in faster training times.

4.5.3 Learning rate and optimization techniques. We show empirical experiments with various optimization algorithms and learning rates, summarized in Fig. 5b. Surprisingly, we see that SGD with exponential learning rate decay outperforms most optimizers with sophisticated learning rate adaptations. We believe this is due to large variance and overfitting in the early stages of training. These issues have been observed in other tasks such as [8, 32], suggesting the need to use tricks such as warm-up heuristics when using momentum-based optimization algorithms to learn embeddings on large, diverse datasets such as ours.



(a) Effect of negative sampling on prediction. Higher number of negative samples results in faster training times.



(b) Various optimization algorithms and learning rates. Sophisticated momentum methods seem to overfit to the early batches too quickly.

Fig. 5. Effect of negative sampling and optimization methods.

5 ONLINE A/B TESTS

We performed online tests on search ranking in order to evaluate the embeddings. Specifically, we use the embeddings as input features in the ranking model. The model implements a neural architecture and takes as input search features (destination, dates, number of travelers, etc.) and property features (price, geographical information, ratings, etc.). The model is trained in the context of Learning to Rank with a pairwise loss approach and the main off-line metric is NDCG.

We performed two tests in order to validate the effectiveness of the embeddings. In the first test, we compared the ranking model with hotel embeddings learned with the MF approach (described in Section 4) against the ranker that was trained without hotel embeddings. This was our initial version of hotel embeddings.

In the next test we compared the ranker model trained with hotel2vec embeddings and compared against the ranker that leverages MF embeddings. Note that these tests were ran in a sequential manner because the two approaches have been designed one after the other.

Tables 5a and 5b present the results of the tests in terms of the main metrics, that is conversion rate (CVR) and gross profit (GP). The first test had no effect in terms of CVR while it was significantly improving GP[3]. Our analysis showed that this was due to the fact that the embeddings would favor better quality hotels which are slightly more expensive.

Table 5b shows the results of the second online test in terms of CVR and GP uplift. In this case we have observed a positive uplift in terms of CVR while not hurting GP. Hotel2vec is able to capture better similarities and as a consequence would help the ranker to propose properties with higher utility for the user. Also, the fact that we could impute hotels that were not seen during training of hotel2vec had a positive effect as we observed an uplift on the main metrics for new and less popular hotels.

[3]Disclosure of specific numbers was not allowed by Legal Department.

After the tests were completed, as the ranking model is linear, we also looked at feature importances on logged searches in order to understand the different behavior. We found out that hotel2vec features were 4 times more important than the MF approach. This result reinforced the conclusion that hotel2vec can capture better similarities among hotels which are later leveraged by the ranker in order to propose higher utility hotels for the user.

Table 5. Results of online tests.

(a) MF embeddings approach.

Metric	CVR	GP
Uplift	Neutral	Positive

(b) hotel2vec approach.

Metric	CVR	GP
Uplift	Positive	Neutral

6 CONCLUSION

In this work, we propose a framework to learn a semantic representation of hotels by jointly embedding hotel click data, geographic information, user rating, and attributes (such as stars, whether it has free breakfast, whether pets are allowed, etc.). Our neural network architecture extends the skip-gram model to accommodate multiple features and encodes each one separately. We then fuse the sub-embeddings to predict hotels in the same session. Through experimental results, we show that enriching the neural network with supplemental, structured hotel information results in superior embeddings when compared to a model that relies solely on click information. Our final embedding is composed from the stack of multiple sub-embeddings, each encoding the representation for a different hotel aspect, resulting in a modular representation. It is also adaptive, in a sense that if one of the attributes or user ratings changes for a hotel, we can feed the updated data to the model and easily obtain a new embedding. Although we mainly focus on learning embeddings for hotels, the same framework can be applied to general item embedding, such as product embedding on Amazon, Ebay, Netflix, or Spotify.

ACKNOWLEDGMENTS

The authors would like to thank Ion Lesan, Peter Barszczewski, Daniele Donghi, and Ankur Aggrawal for helping us collecting hotel's attribute, click and geographical data. We would also like to thank Dan Friedman and Thomas Mulc for providing useful comments and feedback.

REFERENCES

[1] Martin Abadi, Ashish Agarwal, Paul Barham, and et. al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Jens Adamczak, Gerard-Paul Leyson, Peter Knees, Yashar Deldjoo, Farshad Bakhshandegan Moghaddam, Julia Neidhardt, Wolfgang Wörndl, and Philipp Monreal. 2019. Session-Based Hotel Recommendations: Challenges and Future Directions. *arXiv preprint arXiv:1908.00071* (2019).

[3] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Raghu Ramakrishnan. 2013. Content recommendation on web portals. *Commun. ACM* 56, 6 (2013), 92–101.

[4] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 1–6.

[5] Mostafa Bayomi, Annalina Caputo, Matthew Nicholson, Anirban Chakraborty, and Sèamus Lawless. 2019. CoRE: a cold-start resistant and extensible recommender system. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 1679–1682.

[6] Hugo Caselles-Dupré, Florian Lesaint, and Jimena Royo-Letelier. 2018. Word2vec Applied to Recommendation: Hyperparameters Matter. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada) *(RecSys '18)*. Association for Computing Machinery, New York, NY, USA, 352–356. https://doi.org/10.1145/3240323.3240377

[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. ACM, New York, NY, USA, 191–198. https://doi.org/10.1145/2959100.2959190

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

[9] Kallirroi Dogani, Matteo Tomassetti, Sofie De Cnudde, Saúl Vargas, and Ben Chamberlain. 2019. Learning Embeddings for Product Size Recommendations. (2019).

[10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.

[11] Mihajlo Grbovic and Haibin Cheng. 2018. Real-Time Personalization Using Embeddings for Search Ranking at Airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 311–320. https://doi.org/10.1145/3219819.3219885

[12] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1809–1818.

[13] Rachid Guerraoui, Erwan Le Merrer, Rhicheek Patra, and Jean-Ronan Vigouroux. 2017. Sequences, items and latent links: Recommendation with consumed item packs. *arXiv preprint arXiv:1711.06100* (2017).

[14] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 297–304.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf

[16] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. 2020. Contextual and Sequential User Embeddings for Large-Scale Music Recommendation. In *Fourteenth ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 53–62. https://doi.org/10.1145/3383313.3412248

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*. 1026–1034.

[18] Peng Hu, Rong Du, Yao Hu, and Nan Li. 2019. Du., R.: Hybrid item-item recommendation via semi-parametric embedding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*. 10–16.

[19] Xiaowen Huang, Shengsheng Qian, Quan Fang, Jitao Sang, and Changsheng Xu. 2018. Csan: Contextual self-attention network for user sequential recommendation. In *Proceedings of the 26th ACM international conference on Multimedia*. 447–455.

[20] Schockaert Jeawak, Jones. 2018. Embedding Geographic Locations for Modelling the Natural Environment using Flickr Tags and Structured Data. *arXiv preprint arXiv:1810.12091* (2018).

[21] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[22] Buket Kaya. 2019. Hotel recommendation system by bipartite networks and link prediction. *Journal of Information Science* (2019), 0165551518824577.

[23] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30–37. https://doi.org/10.1109/MC.2009.263

[24] Adam Lerer, Ledell Wu, Jiajun Shen, Timothée Lacroix, Luca Wehrstedt, Abhijit Bose, and Alexander Peysakhovich. 2019. PyTorch-BigGraph: A Large-scale Graph Embedding System.. In *Proceedings of the 2nd SysML Conference*.

[25] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*. ACM, 31–40.

[26] Gengchen Mai, Krzysztof Janowicz, Bo Yan, Rui Zhu, Ling Cai, and Ni Lao. 2020. Multi-Scale Representation Learning for Spatial Feature Distributions using Grid Cells. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJljdh4KDH

[27] Leland McInnes, John Healy, and James Melville. 2020. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv:1802.03426 [stat.ML]

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[29] Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. *Rethinking Item Importance in Session-Based Recommendation*. Association for Computing Machinery, New York, NY, USA, 1837–1840. https://doi.org/10.1145/3397271.3401274

[30] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[31] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[32] Martin Popel and Ondřej Bojar. 2018. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics* 110, 1 (2018), 43–70.

[33] Barbara Rychalska, Piotr Bąbel, Konrad Gołuchowski, Andrzej Michałowski, and Jacek Dąbrowski. 2021. Cleora: A Simple, Strong and Scalable Graph Embedding Scheme. arXiv:2102.02302 [cs.LG]

[34] Loveperteek Singh, Shreya Singh, Sagar Arora, and Sumit Borar. 2019. One Embedding To Do Them All. *arXiv preprint arXiv:1906.12120* (2019).

[35] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-pointer co-attention networks for recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2309–2318.

[36] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 225–232.

[37] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. 2018. Attention-based transactional context embedding for next-item recommendation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[38] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.

[39] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Modeling Complementary Products and Customer Preferences with Context Knowledge for Online Recommendation. In *WSDM, 2020*.

[40] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. 2019. Multi-order attentive ranking model for sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5709–5716.

[41] Shuai Yu, Yongbo Wang, Min Yang, Baocheng Li, Qiang Qu, and Jialie Shen. 2019. NAIRS: A neural attentive interpretable recommendation system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 790–793.

[42] Angelina Ziesemer and J Oliveira. 2011. How to know what do you want? a survey of recommender systems and the next generation. In *Proceedings of the Eighth Brazilian Symposium on Collaborative Systems, SBSC*. 104–111.