

# Higher-Order DeepTrails: Unified Approach to \*Trails

Tobias Koopmann<sup>1</sup>, Jan Pfister<sup>1</sup>, André Markus<sup>2</sup>, Astrid Carolus<sup>3</sup>, Carolin Wienrich<sup>2</sup>  
and Andreas Hotho<sup>1</sup>

<sup>1</sup>University of Würzburg, Department of Computer Science, CAIDAS, Chair for Data Science, Germany

<sup>2</sup>University of Würzburg, Institute Human-Computer-Media, Psychology of Intelligent Interactive Systems, Germany

<sup>3</sup>University of Würzburg, Institute Human-Computer-Media, Media Psychology, Germany

## Abstract

Analyzing, understanding, and describing human behavior is advantageous in different settings, such as web browsing or traffic navigation. Understanding human behavior naturally helps to improve and optimize the underlying infrastructure or user interfaces. Typically, human navigation is represented by sequences of transitions between states. Previous work suggests to use hypotheses, representing different intuitions about the navigation to analyze these transitions. To mathematically grasp this setting, first-order Markov chains are used to capture the behavior, consequently allowing to apply different kinds of graph comparisons, but comes with the inherent drawback of losing information about higher-order dependencies within the sequences. To this end, we propose to analyze entire sequences using autoregressive language models, as they are traditionally used to model higher-order dependencies in sequences. We show that our approach can be easily adapted to model different settings introduced in previous work, namely HypTrails, MixedTrails and even SubTrails, while at the same time bringing unique advantages: 1. Modeling higher-order dependencies between state transitions, while 2. being able to identify short comings in proposed hypotheses, and 3. naturally introducing a unified approach to model all settings. To show the expressiveness of our approach, we evaluate our approach on different synthetic datasets and conclude with an exemplary analysis of a real-world dataset, examining the behavior of users who interact with voice assistants.

## Keywords

Behavior Analysis, Sequential Data Analysis, Autoregressive Language Models

## 1. Introduction

Understanding and describing human behavior by analysing transitions between different actions or states has been an established field of research for several years now. It aims to study the dynamics of human behavior by analyzing sequences of user transitions over different states and applying sequential analysis techniques. Understanding human behavior and identifying the most common patterns of interaction can lead to improvements in many aspects, for example, web site design, traffic routing, or usability of different devices. As an exemplary use case, we will dive into the analysis of interactions with digital voice assistants like Alexa or Google Home. These smart devices have become increasingly popular in households over the last few years, capturing and responding to voice commands, aiming to help users with their daily tasks.

LWDA'23: *Lernen, Wissen, Daten, Analysen*. October 09–11, 2023, Marburg, Germany

✉ [koopmann@informatik.uni-wuerzburg.de](mailto:koopmann@informatik.uni-wuerzburg.de) (T. Koopmann); [pfister@informatik.uni-wuerzburg.de](mailto:pfister@informatik.uni-wuerzburg.de) (J. Pfister); [andre.markus@uni-wuerzburg.de](mailto:andre.markus@uni-wuerzburg.de) (A. Markus); [astrid.carolus@uni-wuerzburg.de](mailto:astrid.carolus@uni-wuerzburg.de) (A. Carolus); [carolin.wienrich@uni-wuerzburg.de](mailto:carolin.wienrich@uni-wuerzburg.de) (C. Wienrich); [hotho@informatik.uni-wuerzburg.de](mailto:hotho@informatik.uni-wuerzburg.de) (A. Hotho)



© 2023 by the paper's authors. Copying permitted only for private and academic purposes.



CEUR Workshop Proceedings (CEUR-WS.org)

Sequences of usage behavior, if systematically analyzed, can offer valuable insight into the behavioral patterns, and therefore help improve the usability of the device.

To mathematically represent these sequences, one approach is to aggregate the sequences into graph-like structures with respective transitions between states. Based on this, approaches have been proposed which rely on first-order Markov chain models, such as HypTrails [2], MixedTrails [3] and SubTrails [4]). Hypotheses represent intuitions about human behavior and are constructed and ranked according to how well they fit the observed data. We argue that this aggregation does not come without limitations: mainly the usage of first-order Markov chains is unable to capture vital information about the sequence, like higher-order dependencies. Real-life user behavior is seldom first order; consequently, we propose to model behavior explicitly as sequences and show that allowing for higher-order dependencies by default is a natural fit for this setting [1].

We propose to leverage recent advances in machine learning approaches to address this setting while being able to naturally capture higher-order dependencies in human behavior. For this, the natural choice are autoregressive language models, commonly used in Natural Language Processing. After fitting a model to sequences of user behavior, we propose to test the “validity” of a hypothesis for the training data by evaluating the model’s loss. This effectively determines whether the hypotheses exhibit expected behavior with respect to the observed user actions. Thereby we introduce an explicitly sequence-aware variation to HypTrails, MixedTrails, and SubTrails. The latter is a setting without available hypotheses, where we show how to incorporate transition features to analyze the sequences in a self-supervised manner.

Being able to model higher-order dependencies within user interactions provides valuable insights into user behavior patterns and decision-making processes, consequently surpassing the expressiveness of previous approaches. The insights derived from this research have implications for improving user experience, personalizing recommendations, and designing more intuitive and adaptive systems.<sup>1</sup>

## 2. Related Work

Our work is located in the intersection of two research areas: firstly user behavior analysis from sequences or graph-structured data and secondly sequential machine learning architectures.

**User Behavior Analysis** describes the research domain of analyzing human behavior in any kind of sequences or graphs. The most closely related work uses hypotheses about human behavior to evaluate to which degree a certain hypothesis fits the observed transitions. Namely HypTrails [2] uses Bayesian inferences and sets a prior according to the believed transition probabilities from the hypothesis. The marginal likelihood for each hypothesis with respect to the observed data is calculated, and thus, the hypotheses can be ranked according to how well they fit the observed user behavior. MixedTrails [3] analyzed heterogeneous data, allowing researchers to study sequential data with varying behaviors. Here, each transition is manually assigned to a group, and each group can be explained with its own hypothesis. Furthermore, Subtrails [4] proposes a method to detect interpretable subgroups with exceptional transition

---

<sup>1</sup>Our source code is available at <https://github.com/LSX-UniWue/DeepTrails>.

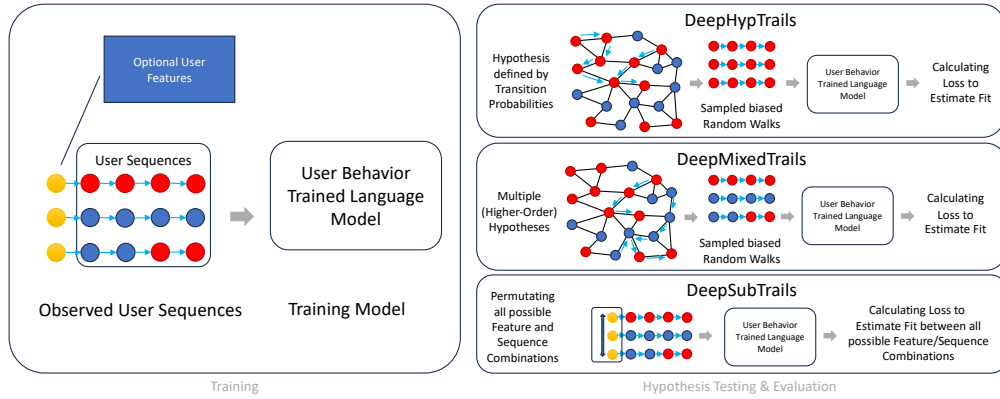
behavior from sequential data. These hypothesis-driven approaches were also adapted on multigraphs [5] by creating a first-order Markov chain from the multigraph instead of aggregated sequences. Finally, behavioral networks can be compared using commonly used graph metrics such as centrality, graph distance, and number of triangles [6]. All of these approaches aggregate the sequences to first-order Markov chains, and hence lose information about higher-order dependencies.

**Machine Learning for Sequential Data** has been a challenging setting, primarily due to the temporal dependencies present in the data. In comparison, traditional machine learning models, such as Random Forest [7] or Support Vector Machines [8], are powerful but also limited to handling data with fixed-length feature vectors. Nowadays, sequential data is usually processed using the transformer architecture [9]. Based on this architecture, different forms of autoregressive language models were developed [10, 11, 12], which are commonly used in Natural Language Processing, where the long-range and higher-order dependencies of words and tokens are a relevant topic. Due to their effectiveness, sequential language models have also been adapted in other areas of research, where it might not seem intuitive at first: e.g. in the research domain of recommendation [13, 14], but also graph-based machine learning approaches started by embedding nodes using sequential random walks and a form of Word2Vec [15, 16, 17].

### 3. Methodology

This work introduces a novel methodology to analyze and describe sequential user behavior. For this, we follow established settings as introduced in HypTrails and its follow-up extensions. Given a set of user observations modeled as sequences, the goal is to either find the best matching hypothesis that explains the observed user behavior (HypTrails [2] & MixedTrails [3]) or to find “interesting” subgroups of users that behave differently from other groups (SubTrails [4]). These existing approaches address this topic by limiting themselves to analyzing single-step transition behavior, hence breaking the observed sequences into first-order Markov chains and analyzing these using Bayesian inference. We argue that this inherently fails to take into account the sequential nature of the data and therefore propose using sequential machine learning models to address this problem. Specifically, we use autoregressive language models, traditionally applied to Natural Language Modeling and sequential data, based on the intuition that the models will discover and utilize higher-order dependencies.

The following sections explain how we model user behavior represented as sequences with autoregressive language models, as well as measure how well a (higher-order) hypothesis matches these user sequences. In addition to these HypTrails [2] and MixedTrails [3] settings, we also explore a setting without available hypotheses to show how to take advantage of transition features to analyze the sequences in a self-supervised manner (cf. SubTrails [4]). We can address all these settings using our language model-based approach with only minor modifications needed between the settings, as depicted in Figure 1 and described in the following. We begin by introducing our common underlying methodology in Sections 3.1 and 3.2.



**Figure 1: DeepTrails:** Schematic overview of our approach. We train a small language model on observed user sequences, optionally with user features. Then we freeze the trained model and plug it into the respective setting: 1. DeepHypTrails: evaluating the model on sequences generated from first-order hypothesis, or 2. DeepMixedTrails: using sequences that contain mixed transition behavior, or 3. DeepSubTrails: identifying interesting subgroups of features.

### 3.1. Representation of User Behavior Using Autoregressive Language Models

Traditionally autoregressive language models are trained to predict the next token given the sequence of previous observations. Mathematically, a language model can be described by the probability assigned to a sentence  $x$ , defined as the conditional probability over all next words  $s_t$  given all previous words  $s_{<t}$  [18]:  $P(x) = \prod_{t=1}^T P(s_t | s_{<t})$ , where  $s_t$  is the input token at time step  $t$ . We aim to exploit the fact that this is closely related to modeling sequences of user behavior by training autoregressive language models on these sequences to describe and analyze user behavior. To this end, we model user states as input tokens by mapping every distinct state  $s$  of a user’s behavior to its own token, thus generating our state-vocabulary  $S$ . Using teacher-forcing [9] and the cross-entropy loss, we train the model to predict the next state given the sequence of all previous states (Figure 1). The model thus learns the transition probabilities for the current state  $s_t$  given the entire history of previously visited states  $s_{<t}$ .

### 3.2. Estimating How Well Hypotheses Align with the Training Data

The core of our methodology is to measure how “surprised” the trained and frozen model is, when given a sequence of user behavior not seen during training. The intuition is that a model trained on sequences of specific user behavior can adequately model sequences that were created by the same behavior, while unseen behavior is likely to be surprising to the model. Therefore, we can identify the types of behavior the model encountered during training and those it did not.

We measure how surprised the model is when presented with a new sequence using the loss function. For this, we freeze the model, and we calculate the positionwise cross-entropy loss of our model, between the sequence and the model’s prediction, like during training, but we do not update the model. By calculating the mean loss over the entire sequence, we obtain a measure of the model’s fit for the currently evaluated sequence: a small loss indicates that the model “expected” this sequence, while a large loss shows that the model has not seen similar user

behavior during training. Additionally, we are able to analyze the loss per position, allowing us to provide a more in-depth analysis of the sequences, which helps us to identify where a hypothesis might be lacking - something that is not possible using the traditional HypTrails methodology.

### 3.3. DeepHypTrails: Evaluating sequences on Homogeneous Observations

HypTrails [2] aims to analyze sequential user behavior using hypotheses. The setting is to identify the *best matching* hypothesis for the observed data, that is, the hypothesis that best describes the observed sequences. Since HypTrails only allows first-order dependencies, hypotheses are represented as  $|S| \times |S|$  transition probability matrices, which describe the transition probabilities from one state to any other given state.

In contrast, our approach uses entire sequences, which we can generate by sampling biased random walks on the transition probability matrices (Figure 1). We then evaluate our model on these randomly generated walks as described in Section 3.2. By averaging over all random walks - each following a specific hypothesis - we identify the hypothesis resulting in the lowest loss.

### 3.4. DeepMixedTrails: Analyzing Hypotheses on Heterogeneous Observations

This extension of HypTrails allows the analysis of heterogeneous sequences. The underlying assumption is that the sequences are not generated by a single driving force but by multiple driving forces. Therefore, it is assumed that transition probabilities can change given the circumstances, e.g. after a certain amount of time, or they might even originate from entirely different groups of users. Since the method proposed in MixedTrails [3] is strictly transition-based and relies on a first-order Markov chain, it inherently cannot distinguish if sequences 1. change behavior after a certain amount of time, i.e. contain higher-order dependencies, or 2. originate from different groups, i.e. consist of different types of first-order behavior. Each case can be exemplary illustrated as a soccer team that plays offensive in the first half and defensive in the second or tourists and locals that have a different movement behavior in a city [3].

Since our approach takes entire sequences into account, it is naturally able to distinguish between these two settings. The model has no access to information on which behavior a sequence is generated by. We rely entirely on the model to implicitly learn the different behaviors present on its own, such that the loss is low when testing hypotheses that match the training sequences, and high if the behavior was not present in training (section 3.2). Additionally, this even allows us to combine both scenarios of MixedTrails, by training the model on sequences from different groups, where each might contain higher-order dependencies.

### 3.5. DeepSubTrails: Identifying Subgroups with Interesting Behavior

Lastly, we use our approach in the SubTrails setting [4]. For this, instead of using predefined hypotheses to describe human behavior, we analyze observed user behavior to find unique subgroups that show “exceptional” transition behavior. These subgroups exhibiting interesting transition behaviors - when compared to all transitions - are identified by attributes or features assigned to each transition. We have to slightly adapt our approach to accommodate for this

scenario, by conditioning our autoregressive model on the features that are associated with the sequence (details in Section 4.3). To this end, we feed all features to the model, and thus enable it to learn a different conditional behavior based on the given feature expressions. For evaluation, we assess the loss across all possible combinations between features and observed sequences, thereby identifying interesting user behavior through the corresponding loss values. A large loss indicates that the given combination of feature expression and sequence is unexpected to the model and, respectively, shows atypical sequential behavior for this feature expression. Furthermore, we introduce a similarity measure for features: features are similar if the model assigns a similar loss for the same sequences, when conditioned with the respective feature. This measure allows us to cluster similar feature expressions together, and thus identify similarly behaving users. As before, our approach has the advantage of naturally capturing higher-order dependencies.

## 4. Experimental Setup

We create several different artificial datasets with known synthetic user behavior. Additionally, we demonstrate the expressiveness of our approach on a real-world dataset, in which we analyze the behavior of users interacting with voice assistants.

### 4.1. Generation of Synthetic Datasets

We generate several synthetic datasets containing different types of user behavior that can be described or uncovered using our methodology introduced in Section 3. To artificially generate user transitions, we define user behavior over an underlying Barabasi-Albert graph [19]. This graph structure is scale-free, which means that the graph is densely connected in the “center” and less connected in the periphery. For all synthetic datasets, we generate a graph with  $n = 100$  nodes and new nodes connected to  $m = 10$  existing nodes using preferential attachment [19]. We divide the graph nodes equally into *even* and *odd* nodes, allowing us to define synthetic user behavior based on these classifications. To model different types of behavior, we introduce differently biased random walkers that follow a predefined transition behavior based on the node categories. For each synthetic behavior, we start a new biased random walker from each node 1000 times and generate sequences of length 20. Notably, the model has no explicit information about the underlying graph structure or node categorization and only has access to the exhibited user observations.

**Synthetic Data for HypTrails** To evaluate our approach, we define different synthetic behaviors as follows: (i) **even**: The walker only transitions towards even nodes (ii) **odd**: The walker only transitions towards odd nodes (iii) **random**: The walker randomly transitions towards any adjacent node (iv) **teleport**: In contrast the walker randomly teleports to any node on the graph. Additionally, to show the applicability of our model to noisy data, we create matching biased probabilistic walkers for each behavior, where the walker follows the hypothesis only 90% of the time, otherwise following the opposite behavior. Following each of the above behaviors, we generate separate sets of sequences for observed user behavior as well as for hypotheses, which will be used for evaluation.

**Synthetic Data for MixedTrails** As explained in Section 3.4 we distinguish between two scenarios (i) The sequences are generated by different driving forces or (ii) transitions within a sequences are created by a changing behavior. The latter scenario can be modeled by using higher-order dependencies in the creation process. We create a single dataset containing both scenarios at the same time: a mixture of sequences originating from several differently biased random walkers, and some of them even showing varying behavior over time. To this end, we add the following behaviors: (i) **first-even**: The walker only transitions to even nodes for the first half of the walk, and only odd nodes for the second half. (ii) **first-odd**: The walker only transitions as above, but in reverse. (iii) **two-odd-two-even**: The walker transitions twice to odd nodes, then twice to even nodes, and so forth. The dataset consists of 330 walks per node for each of the following behaviors: `even`, `odd` and `first even`.

**Synthetic Data for SubTrails** Following the SubTrails approach, we only generate observed sequences and no hypotheses, but introduce additional matching feature vectors for each sequence. The model has access to this feature vector consisting of attributes that *may* influence user behavior. Some features explicitly correlate with the observed behavior, while some do not.

Here, our dataset consists of 250 biased random walks per node each: `even`, `odd`, `first even`, and `first odd` bias. We add a feature vector with six binary features, where the first feature activates only if the random walker follows the even bias, the second feature only activates if the random walker follows an odd bias, etc. The last two features are activated at random and serve as noise that the model has to learn to ignore. This results in a total of 16 possible feature combinations, where the last two features exhibit four potential permutations. The first 4 features are one-hot encoded, thereby combining the 4 permutations to each one-hot encoding, we can create 16 possible feature combinations.

## 4.2. Case Study on Long-Term Study on Voice Assistant Usage

As a real-world case study of our methodology, we analyze a dataset containing various types of user behavior. It was created by a long-term study in which 39 students received an Amazon Alexa or Google Home device and their usage was tracked and analyzed. For our analysis, we represent user interactions with these devices as sequences. To this end, we introduce 33 distinct states that are shared between all sequences. Every state represents one type of command, e.g. “playing music” or “asking about the news”. Sequences are constructed from consecutive voice commands: Every time a user interacts with the device twice within a 15-minute time window, a transition between the two types of voice commands is added for the current sequence. Thus, we construct 217 sequences with a length between 2 and 94 and an average of 6.43. Information about the users’ perception of the voice assistants as well as other psychological features have been systematically collected using questionnaires. We use these features, e.g. how lonely a user is or to which extent the user describes the voice assistant as a friend as additional input for the model. We also use time of day and day of the week as additional metadata, leading to 150 distinct feature sets, each used by at least one user.

### 4.3. Autoregressive Language Models

Our methodology requires a model that can be used to autoregressively model sequences. Traditional language models are an obvious fit, but we also explore a significantly smaller and deterministic language model based on a Random Forest.

**Transformer Decoder** The first model uses a transformer decoder following the GPT architecture. Every state in the sequence is modeled as a token. The model is trained to predict the next state given the history of current states autoregressively using teacher forcing [9]. We use the nanoGPT implementation<sup>2</sup> with a vocabulary size of “number of states” plus 2 (e.g. 100 nodes + EOS and BOS tokens for the synthetic datasets), four layers with four heads, and an embedding dimension of 16. For our DeepSubTrails setting, we additionally adapt the model to encode a feature vector. Categorical features are encoded as a one-hot vector, whereas numerical features are kept as is. All features are concatenated into a vector and then consequently embedded to match the dimensionality of the token embedding. This new token embedding replaces the BOS token, thus conditioning the model to predict a sequence with respect to the given user features. This is comparable to user embeddings in sequential recommendation [20].

**Random Forest-based Language Model** As a second model, we will use a language model based on a random forest classifier and train it on our sequences. For this, we encode each state as a one-hot vector and multiply it by an exponentially decaying weight, depending on the positional distance of the embedded token to the current token. This can be compared to a positional embedding. All these weighted one-hot encoded vectors are summed up into a single vector, and thus the shape becomes independent of the sequence length. The positional information for each state is encoded in the magnitude of this multi-hot vector. For modeling user features, we create a vector which contains an integer per categorical feature and a float per numerical value, and concatenate it to the sequence information described above. Thus, the model has access to the sequence and user information at the same time.

## 5. Results

In the following section, we will show the different experiments conducted, interpret the results, and show further analysis with respect to the learned embeddings. We conduct experiments for all of the previously mentioned scenarios, namely DeepHypTrails, DeepMixedTrails, and DeepSubTrails, and additionally show the applicability of our approach in a real-world setting, where users interact with a voice assistant.

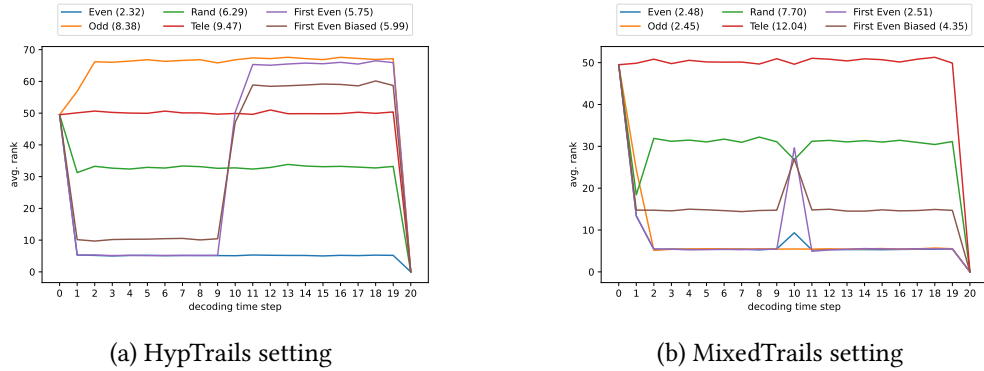
### 5.1. DeepHypTrails

In the following, we evaluate the GPT model in the DeepHypTrails setting on synthetic data as previously introduced. The Random Forest language model leads to similar results, but we limit our analysis here to GPT due to space constraints; additional plots for the random forest model

---

<sup>2</sup><https://github.com/karpathy/nanoGPT>





**Figure 2:** Applying autoregressive language models, here GPT, to sequential behavior analysis. Figure 2a shows the results trained on sequences with only even behavior and evaluates it on different hypotheses. For each time step, we rank the predicted tokens by logits in descending order. We plot the average rank of the target token per decoding step for each hypothesis. A low rank indicates a high likelihood for the next transition according to the model. Figure 2b shows the result when training on sequences of heterogeneous behaviors, as introduced in Section 4.1. The number in the legend indicates the average loss over all sequences and positions for the given hypothesis.

can be found in the appendix<sup>3</sup>. Figure 2a shows the rank-wise analysis for different hypotheses of the output of our model, when trained on even behavior. The x-axis shows the decoding time step, i.e. the position in the walk. At each time step, we rank the tokens according to their logits in descending order. The y-axis shows the average rank of the target token per position, where a low rank indicates a high likelihood for the next token according to the model. All hypotheses begin with an average rank of  $\sim 50$ , since we start from each of the 100 nodes in the graph equally often. As the model has to randomly guess the first node from which the walk starts, the expected average rank for this step is 50. Furthermore, the last node at decoding step 20 has an average rank of 1, since all synthetic sequences have a length of 20 and the model is able to predict the position of the EOS token perfectly. Sequences created from hypotheses showing the same behavior as the observed sequences have the lowest average ranks, as expected and shown by the blue line (even). The opposing behavior (orange line, odd) leads to average ranks above 60, which shows that the model expects these nodes to be very unlikely. Using a random walker (green line, rand) as hypothesis leads to an average rank of  $\sim 30$ , which is higher than a hypothesis with even bias, but also lower than the odd walker. Naturally, transitioning at random is more similar to the observed user behavior than actively following an opposite behavior than during training. Furthermore, a teleporting walker (red line, tele) stays at an average rank of  $\sim 50$  after the first step, which is expected, since this again means that the model cannot predict the next transition. Furthermore, this shows the ability of the model to learn the graph structure, since the average rank of the rand hypothesis is lower, showing that the model is less surprised by a randomly sampled adjacent node than any randomly sampled node. Finally, using sequences with higher-order dependencies in Figure 2b, namely first even, we can observe a similar rank for the first half of the sequences. Subsequently, the average

<sup>3</sup>Appendix can be found on <https://professor-x.de/deeptrails-arxiv>.

rank increases to the same level as the odd hypothesis after the behavior changes. We find that `first even` biased results in comparable ranks, but with a smaller amplitude, as it effectively is a mixture of `first even` and `rand`. This shows that our approach is capable of ranking the sequences generated with the respective behavior accordingly and is even possible to analyze sequences by position, thus indicating at which transitions a hypothesis might not be adequately explaining the data. Furthermore, in Figure 3a, we analyze the extracted token embeddings for each state of a model trained on `first even` observations. The UMAP plot with annotated classes shows that the model is able to neatly separate both state types in the token embedding space, indicating that the model is able to understand the different types of nodes. The label “ST” denotes the embeddings of the special tokens.

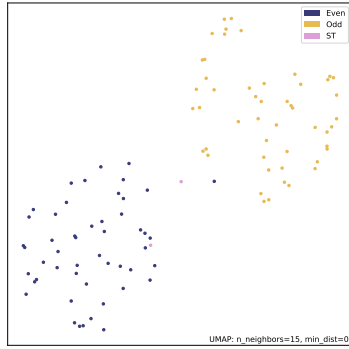
Finally, we show an ablation study in the appendix, where we applied the original HypTrails approach on data created by higher-order sequences. Our results show that HypTrails, due to the use of a first-order Markov chain, is not able to distinguish between the higher-order sequences and random navigation.

## 5.2. DeepMixedTrails

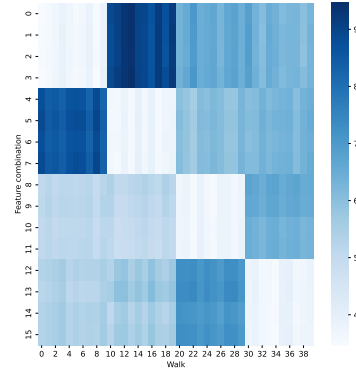
In the next experiment, we use our approach in both MixedTrails settings as introduced in Section 3.4. In Figure 2b, we show the results for a model trained on sequences from different groups of users who also show a change in behavior over time, as explained in Section 4.1. We expect hypotheses that exist in the training observations to have on average a lower rank than hypotheses not occurring in the data. Since the model is trained on equal amounts of `even`, `odd` and `first even` behavior, all of these hypotheses lead to similarly low ranks (see Figure 2b). Notice that the rank at step 1 is about 15, since the model does not know yet to which behavior the current sequence belongs to - either one of the even behaviors follows if it is an odd walk. After time step one this is unambiguous, thus the rank decreases further. At decoding step ten, we can observe a peak which is explainable by higher-order hypotheses `first even`. The model cannot know whether the current hypothesis follows `even` or `first even`, therefore this spike appears for both hypotheses. In particular, this spike is absent for the `odd` hypothesis as the model can be certain to follow this behavior known from the training data. The average rank for the `rand` hypothesis increases after the first decoding step, since the model assumes that the subsequent steps follow the first observed time step and not random nodes. As before, `tele` remains at a rank of ~50. We show additional experiments for DeepMixedTrails in the appendix.

## 5.3. DeepSubTrails

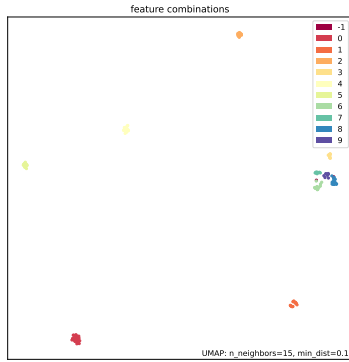
As final synthetic experiment, we analyze transitions with respect to features, instead of using hypotheses. For this, we train our model on the synthetic dataset described in Section 4.1, which contains a feature vector with six categories for each walk. For evaluation, we use 10 exemplary walks per behavior (40 total). Walks 0 to 10 are walks containing `even` behavior, 11 to 20 contain `odd` behavior, 21 to 30 contain `first even` behavior and 31 to 40 contain `first odd` behavior. We sort the matching feature sets in the same ordering, where feature sets 0 to 4 have category 1 activated, all possible permutations for category 5 and 6 (which are walks



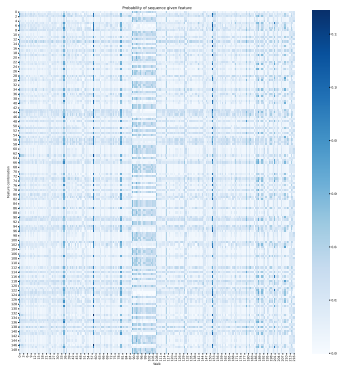
(a) Analysis of Embedding space



(b) Heat map of loss on synthetic data



(c) Feature clustering w.r.t. the probability



(d) Heat map for Voice Assistant dataset

**Figure 3:** Figure 3a shows the ability of our model to learn the graph structure and the different node types. We use a UMAP visualization of node embeddings, which shows a separation between the different node types. Figure 3b shows a visualization of the DeepSubTrails loss scores in synthetic data. The model is trained on sequences prepended with features. For evaluation, we create different combinations of features and walk behavior and visualize the loss. Figure 3d shows the same visualization in our real-world dataset. Here the x-axis lists all command sequences and the y-axis all distinct feature combinations. Figure 3c clusters the possible feature sets with respect to the column-wise probability given by our model from Figure 3d.

containing even behavior), and so forth. We combine the 16 unique feature sets with each of the 40 walks and visualize the resulting loss as a heat map in Figure 3b.

We observe the lowest loss across the diagonal, where the model correctly finds that the behavior in the walks matches the features. The highest loss can be found in the dark blue rectangles in the upper left corners, where the feature suggests even walks, but the walks contain only odd behavior and vice versa. Next, we can observe medium loss scores for the

top right and bottom left corner, where the feature suggests changes in behavior walks, but the walks match only partially by containing only odd or even behavior. Finally, we can observe that noise categories 5 and 6 do not have any impact: the loss is stable across all feature combinations where only these noncorrelating categories change (feature combination 0 to 3, 4 to 7, etc.). Feature combinations 0 through 3 iterate over all possible combinations only of categories 5 & 6, but the loss and the other categories remain stable. By recombining features and sequences, this allows us to estimate how well a feature set matches to each walk and therefore hypothesis and behavior. We add further analysis and visualizations by clustering the features and sequences in the appendix.

#### 5.4. Case Study: DeepSubTrails on Voice Assistant Data

The last experiment uses the DeepSubTrails methodology on our voice assistant dataset, as previously introduced (Section 4.2) using the random-forest based language model. Figure 3d exhibits clear patterns in the data by showcasing which feature set best matches which sequence. We find that several rows possess the same probability distribution across all sequences (columns), showing that these feature sets indicate similar transition behavior. The same phenomenon can be observed when analyzing the heat map column-wise, w.r.t. the sequences: several sequences are similar to each other based on which sets of feature expressions they match best. To analyse this more intuitively, in Figure 3c, we use UMAP and HDBSCAN to cluster similar feature sets by interpreting the scores in each row from the heatmap as its vector representation. Thus, we can identify ten different clusters of feature sets that behave similarly and, therefore, yield similar probabilities for the same sequences. When analyzing these clustered feature sets, we find that commonly different feature sets are clustered together, which differ only by the “day of the week” feature, indicating consistent user behavior across different week days.

Overall we find that there are, in general, two types of clusters: mono-user clusters and multi-user clusters. Six clusters consist of only a single user each, which exhibits the same behavior across different days of the week, but at similar times of the day. For example in cluster ten, we find a cluster of feature sets all stemming from the same user, and only differing by the “day of the week”. The feature sets in the cluster all stem from interactions in the morning and in the associated sequences the user asks the assistant to play music, sometimes in combinations with setting a timer. In the same spirit cluster six consists of a single user who sets *multiple* alarms in the evening, often times in combination with controlling their smart-home and media-playback. This hints at a fixed incorporation of the voice assistant in their bed-time routine. Furthermore cluster two and three consist of different users oftentimes repeatedly asking to play or control their media in the afternoon.

Next, we exemplary analyse the clusters consisting of feature sets stemming from different users. Naturally they are more heterogeneous in their behavior, making interpretation harder. Nevertheless, we find clusters eight and nine, consisting of two users and four users respectively whose voice commands get misunderstood - oftentimes multiple times in a row. This either stems from the device misunderstanding certain song titles and bands, or more commonly: trying to control the device with media controls like “Alexa, next!”, when the current menu (e.g. within a skill) is not controllable with these commands. At the same time, within cluster nine something similar happens but for e.g. Bluetooth control or the calculator. Here it is also

common that no valid or understood command is executed within a sequence, indicating that the user has given up altogether in trying to make the device perform the requested action. Overall this shows the expressiveness of our methodology, as well that it is possible to meaningfully analyse real world datasets with it.

## 6. Conclusion

This work explores the use of autoregressive language models to analyze and describe sequential user behavior. We train a model on observed sequences, therefore, predict user behavior present in these sequences and test hypothesis on user behavior. We evaluate our approach in three different settings, which are adopted from previous work. For the first two settings, we construct (higher-order) sequences, sourced from hypotheses, each embodying a possible explanation for the observed user behavior. Consequently, we calculate the loss of the frozen model on these generated sequences to estimate the fit of the hypothesis for the observed behavior. In a third setting, where user features but no hypotheses are available, we can find exceptional feature sets that indicate unique user behavior. We show the applicability and advantages of our approach on several synthetic datasets and conclude by showing one setting on a real-world dataset, namely the usage behavior of users interacting with their voice assistants.

## Acknowledgements

This work is partially supported by the German Research Foundation (DFG) under grant number 438232455 (HydrAS) and the MOTIV research project funded by the Bavarian Research Institute for Digital Transformation (bidt), an institute of the Bavarian Academy of Sciences and Humanities. The authors are responsible for the content of this publication.

## References

- [1] A. M. Tedesco, F. Bianchini, L. Piccardi, S. Clausi, A. Berthoz, M. Molinari, C. Guariglia, M. Leggio, Does the cerebellum contribute to human navigation by processing sequential information?, *Neuropsychology* 31 (2017) 564.
- [2] P. Singer, D. Helic, A. Hotho, M. Strohmaier, Hyptrails: A bayesian approach for comparing hypotheses about human trails on the web, in: A. Gangemi, S. Leonardi, A. Panconesi (Eds.), *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015, ACM, 2015*, pp. 1003–1013. URL: <https://doi.org/10.1145/2736277.2741080>. doi:10.1145/2736277.2741080.
- [3] M. Becker, F. Lemmerich, P. Singer, M. Strohmaier, A. Hotho, Mixedtrails: Bayesian hypothesis comparison on heterogeneous sequential data, *Data Min. Knowl. Discov.* 31 (2017) 1359–1390. URL: <https://doi.org/10.1007/s10618-017-0518-x>. doi:10.1007/s10618-017-0518-x.
- [4] F. Lemmerich, M. Becker, P. Singer, D. Helic, A. Hotho, M. Strohmaier, Mining subgroups with exceptional transition behavior, in: B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, R. Rastogi (Eds.), *Proceedings of the 22nd ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, ACM, 2016, pp. 965–974. URL: <https://doi.org/10.1145/2939672.2939752>. doi:10.1145/2939672.2939752.

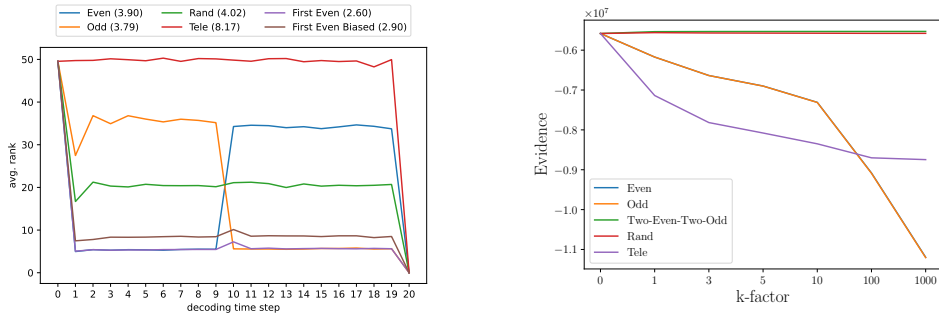
- [5] L. E. Noboa, F. Lemmerich, M. Strohmaier, P. Singer, JANUS: A hypothesis-driven bayesian approach for understanding edge formation in attributed multigraphs, *Appl. Netw. Sci.* 2 (2017) 16. URL: <https://doi.org/10.1007/s41109-017-0036-1>. doi:10.1007/s41109-017-0036-1.
- [6] P. Wills, F. G. Meyer, Metrics for graph comparison: A practitioner’s guide, *CoRR abs/1904.07414* (2019). URL: <http://arxiv.org/abs/1904.07414>. arXiv:1904.07414.
- [7] T. K. Ho, Random decision forests, in: *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, 1995, pp. 278–282 vol.1. doi:10.1109/ICDAR.1995.598994.
- [8] C. Cortes, V. Vapnik, Support-vector networks, *Machine learning* 20 (1995) 273–297.
- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, December 4-9, 2017, Long Beach, CA, USA, 2017, pp. 5998–6008. URL: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>.
- [10] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, *OpenAI blog* (2019).
- [11] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, R. Salakhutdinov, Transformer-xl: Attentive language models beyond a fixed-length context, in: A. Korhonen, D. R. Traum, L. Màrquez (Eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, Association for Computational Linguistics, 2019, pp. 2978–2988. URL: <https://doi.org/10.18653/v1/p19-1285>. doi:10.18653/v1/p19-1285.
- [12] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, Q. V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, in: H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019*, pp. 5754–5764. URL: <https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html>.
- [13] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, P. Jiang, Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, in: W. Zhu, D. Tao, X. Cheng, P. Cui, E. A. Rundensteiner, D. Carmel, Q. He, J. X. Yu (Eds.), *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019, ACM, 2019*, pp. 1441–1450. URL: <https://doi.org/10.1145/3357384.3357895>. doi:10.1145/3357384.3357895.
- [14] W. Kang, J. J. McAuley, Self-attentive sequential recommendation, in: *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018, IEEE Computer Society, 2018*, pp. 197–206. URL: <https://doi.org/10.1109/ICDM.2018.00035>. doi:10.1109/ICDM.2018.00035.

- [15] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: S. A. Macskassy, C. Perlich, J. Leskovec, W. Wang, R. Ghani (Eds.), The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014, ACM, 2014, pp. 701–710. URL: <https://doi.org/10.1145/2623330.2623732>. doi:10.1145/2623330.2623732.
- [16] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, LINE: large-scale information network embedding, in: A. Gangemi, S. Leonardi, A. Panconesi (Eds.), Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015, ACM, 2015, pp. 1067–1077. URL: <https://doi.org/10.1145/2736277.2741093>. doi:10.1145/2736277.2741093.
- [17] M. Ou, P. Cui, J. Pei, Z. Zhang, W. Zhu, Asymmetric transitivity preserving graph embedding, in: B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, R. Rastogi (Eds.), Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, ACM, 2016, pp. 1105–1114. URL: <https://doi.org/10.1145/2939672.2939751>. doi:10.1145/2939672.2939751.
- [18] Y. Bengio, R. Ducharme, P. Vincent, A neural probabilistic language model, in: T. K. Leen, T. G. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA, MIT Press, 2000, pp. 932–938. URL: <https://proceedings.neurips.cc/paper/2000/hash/728f206c2a01bf572b5940d7d9a8fa4c-Abstract.html>.
- [19] A.-L. Barabasi, R. Albert, Emergence of scaling in random networks, Science 286 (1999) 509–512. URL: <http://www.sciencemag.org/cgi/content/abstract/286/5439/509>. doi:10.1126/science.286.5439.509. arXiv:<http://www.sciencemag.org/cgi/reprint/286/5439/509.pdf>.
- [20] E. Fischer, A. Dallmann, A. Hotho, Personalization through user attributes for transformer-based sequential recommendation, in: Workshop on Recommender Systems in Fashion and Retail, Springer, 2022, pp. 25–43.

## 7. Appendix

The appendix contains some additional analysis, which is helpful for a deeper understanding of our work and did not fit in the main manuscript. First, we show some additional experiments for our DeepHypTrails, where we add a model trained on random transitions and one trained on biased observations. Then, we show the second scenario for the DeepMixedTrails approach, where the observed data contain sequences with changing behavior. Furthermore, we will show further analysis for the DeepSubTrails approach by clustering the sequences and features. Finally, we show the same visualizations for our real-world setting on voice assistant commands.

### 7.1. DeepHypTrails: Using biased sequences as observations



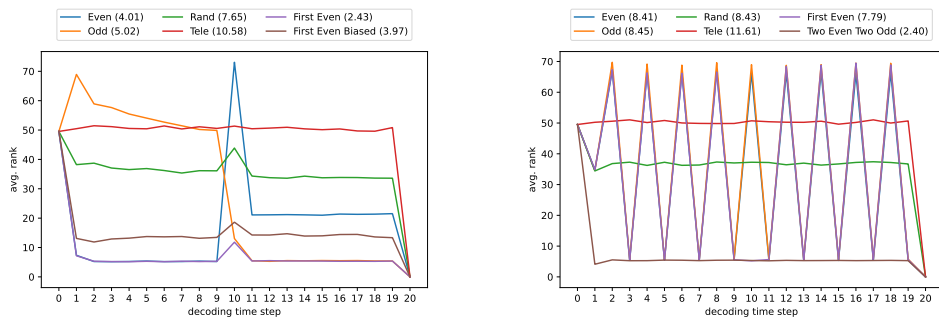
(a) Model trained on biased data, namely first even biased

(b) Applying HypTrails on higher-order sequences.

**Figure 4:** Figure 4a is trained on first-even-biased, which means that the behavior is chosen using 90% and 10% the opposite behavior. Figure 2 shows a HypTrails analysis on higher order sequences. The y-axis shows the evidence (higher is better), and the x-axis the concentration factor. The underlying data were created by the same behaviour as the red hypothesis. We can observe, that the evidence scores for higher order hypothesis (red) and random (purple) are similar. Hence HypTrails is not able to distinguish the higher order dependencies. For a deeper understanding of the plot, we refer to author to [2].

First, we will show some additional experiments for our approach. Figure 4a shows the result of our model trained on first even biased. Meaning 90% of the transitions follow first even behavior and the other 10% the opposite behavior (first odd). We can see, even with noise that the model is able to grasp the underlying behavior. The Hypothesis, which contains the definite, not noisy behavior (purple line, first even) has the overall lowest average rank per position, closely followed by the hypothesis with the same user behavior (brown line, first even biased). For most of the other lines, we can see similar behavior as in Figure 5a, except the phenomenon, that the first half of the orange line (odd behavior) has a higher average rank than the second half of the blue line (even behavior). Intuitively, these lines should be at the same height.





(a) Changing behavior

(b) Using higher-order dependencies

**Figure 5:** Visualisation of DeepMixedTrails second setting, where a model is trained on sequences with the same driving force, but containing higher-order dependencies. Figure 5a shows a model trained on `first even` observations. Figure 5b displays the result when training on observations, which transitions twice to even nodes, then two time to an odd node.

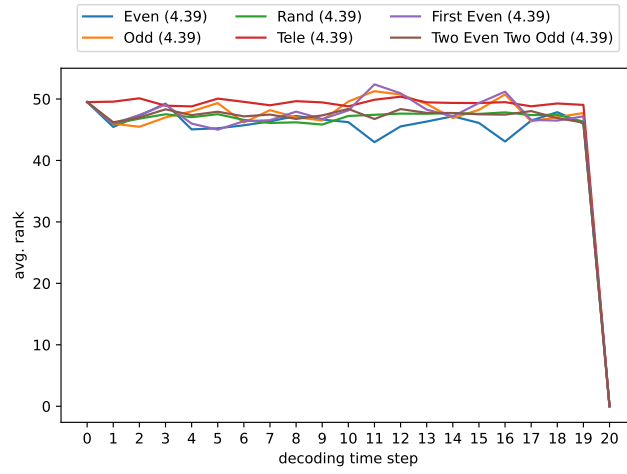
## 7.2. DeepMixedTrails: Higher-order observations

The first scenario of DeepMixedTrails is displayed in Section 3.4 and contains a dataset with different groups of users, each showing different behavior. Now, the second scenario assumes that a sequence contains changing transition probabilities, as seen in Figure 5. Here we demonstrate two scenarios. Figure 5a shows a model trained only on `first even` observations. For this, we trained a model on a single group of users with the same higher-order behavior, namely `first even bias`. We can see that the respective hypothesis `first even` has a low rank across all steps. Hypotheses with first-order dependencies (such as `even` and `odd`) have a low rank for half of the sequence, but not in general - as expected.

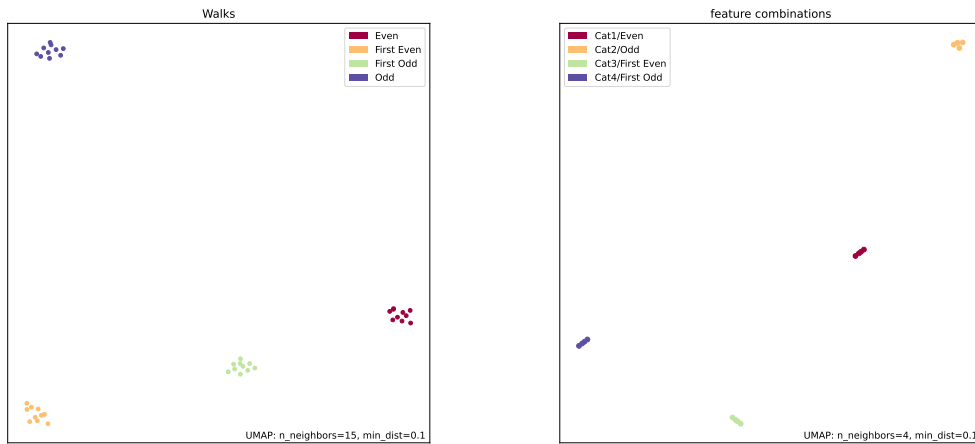
Furthermore, Figure 5b shows another higher-order approach, where during training the biased random walker only transitions to even nodes for two steps and then transitions to odd nodes for another two times. This pattern is then repeated until the end of the sequence. We can see the expected behavior, since the respective hypothesis has a low rank across all steps (purple line). We can observe jumps every two steps for the single modality hypothesis (blue or orange line).

## 7.3. Further Analysis for DeepSubTrails

Furthermore, we show an additional analysis by clustering the loss scores with respect to the features (see Figure 7). The different features, which contain walks with different behaviors, are separable. We can observe the same result, when clustering the sequences, as shown in Figure 7b.



**Figure 6:** Ablation study: Training a model on teleport behavior. As expected, no hypothesis can explain the underlying data.



(a) Clustering the walks

(b) Clustering the features

**Figure 7:** Analysis of features and walks for our synthetic DeepSubTrails dataset. As in the main manuscript for in Figure 3c, we cluster using UMAP and HDBSCAN. We can see that our model is able to distinguish the sequences containing different behavior and the respective cluster neatly.