What next? Modeling human behavior using smartphone usage data and (deep) recommender systems

Master Thesis Presentation

Simon Wiegrebe

October 08, 2021

What next? Modeling human behavior using smartphone usage data and (deep) recommender systems

Simon Wiegrebe

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

1 Introduction

What next? Modeling human behavior using smartphone usage data and (deep) recommender systems

Simon Wiegrebe

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

2 Theoretical Framework

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

2 Theoretical Framework

- early works (big 5)
- uncouple from psych context: app2vec
  - app analogies not very intuitive (example)
  - how to evaluate performance?
- uncouple from psych context, focus exclusively on sequential nature of data
  - time-ordered sequences, imposing session structure
  - similarity to data from movie ratings, e-commerce sessions, social networking sites:
    - several users
    - 1+ sessions per user
    - 1+ events per session
  - use RS models
    - target variable follows a multinomial distribution with a large number of distinct outcomes
    - task is to create a recommendation list
- intrinsic similarity to language data
- session-based and session-aware RS

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

3 Data

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

3 Data

- description (+ table)
- representation and preprocessing: app-level
- representation and preprocessing: sequence-level
- representation and preprocessing: app-to-text conversion

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

4 Methodology

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

4 Methodology

- modeling
  - session-based baseline models
  - session-based neural models
  - session-aware neural models
  - extensions
- evaluation
  - train-validation-test split
  - evaluation protocol
  - evaluation metrics
  - tuning

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

5 Results

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Overall Performance (I)

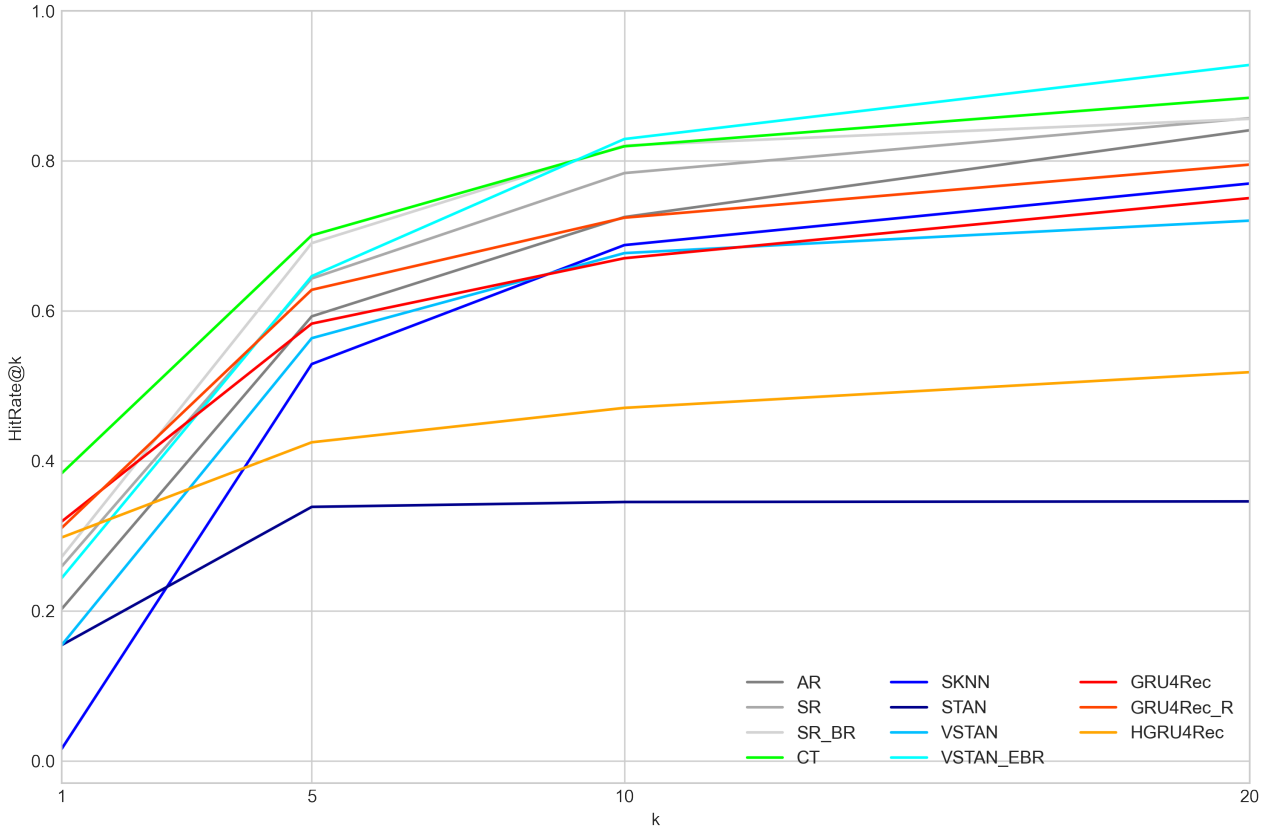

Figure XXX: *HR@k* performance for $k = 1, 5, 10,$ and 20 on five-window app-level data.

What next? Modeling human behavior using smartphone usage data and (deep) recommender systems

Simon Wiegrebe

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Overall Performance (II)

- *CT* has the best $HR@k, \forall k \in \{1, 5\}$, and the best $MRR@k, \forall k \in \{1, 5, 10, 20\}$.
- *VSTAN_EBR* performs best in terms of *HR*@20, i.e., when the Hit Rate is evaluated on the list of the top 20 recommendations per event.
- The three NN-based models *GRU4Rec*, *GRU4Rec_R*, and *HGRU4Rec* come right behind *CT* i.t.o. *HR*@1, yet their relative performance deteriorates as k increases.
- They also have the highest app coverage and the lowest popularity bias.
- Session-aware (heuristic) extensions increase performance of session-based algorithms.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Minimum Sequence Length (I)

- Background:
  - *GRU4Rec*, *GRU4Rec_R*, and *HGRU4Rec* employ RNNs.
  - These learn from the present sequence whereas non-neural methods mostly "look up" similar sequences or app combinations.
  - App-level sequences are typically short $\rightarrow$ RNN-based methods do not have "much to learn from".
- Hypotheses:
  - We expect *GRU4Rec*, *Gru4Rec_R*, and *HGRU4Rec* to perform better on longer sequences, where the underlying RNNs are provided more events to learn from.
  - We expect sequence length to have no impact on performance of *AR*, *SR*, and *SR_BR*, since they only use the last event of a sequence for prediction.
- To test this, we train and evaluate our models on a subset containing only sequences with at least 20 events.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Minimum Sequence Length (II)

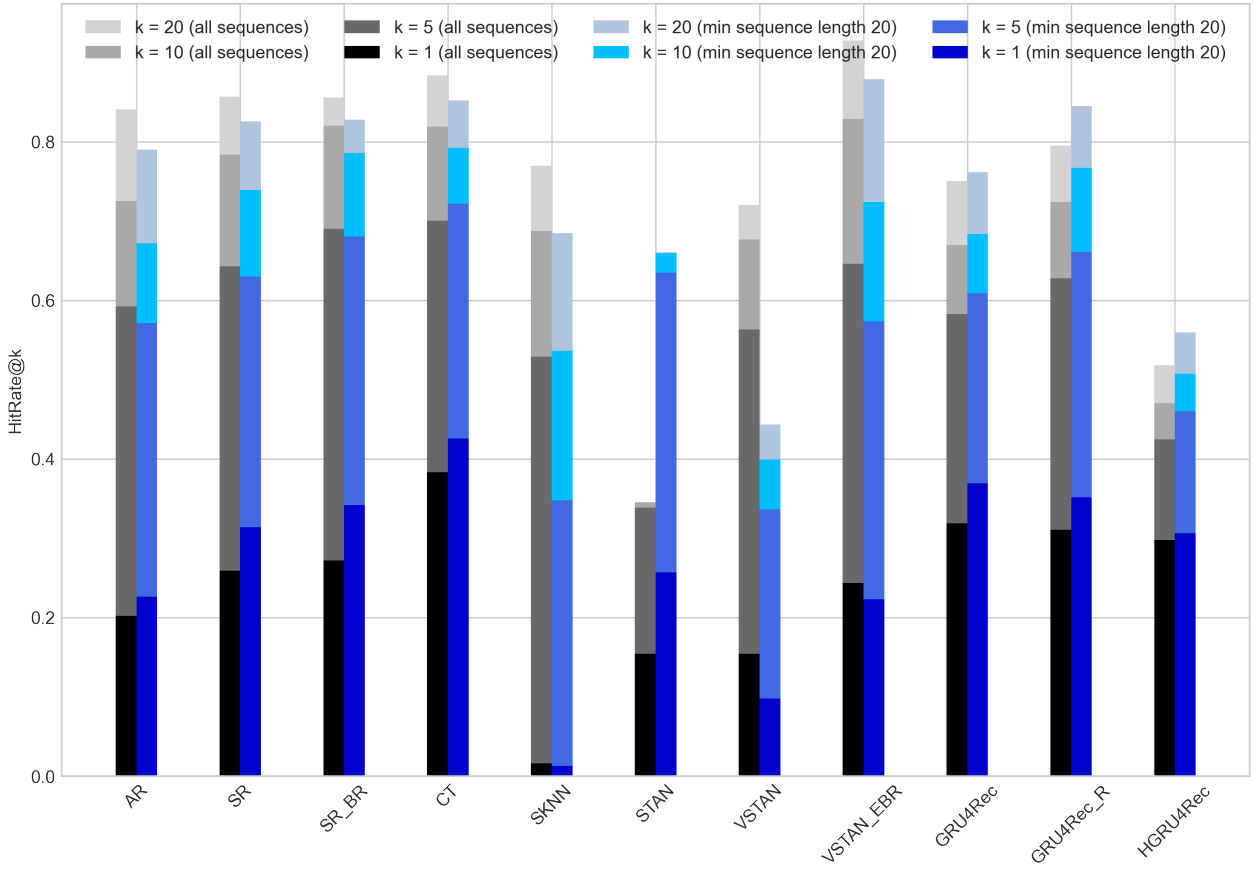

Figure XXX: *HR@k* comparison between performance on full five-window app-level data (left bars) and performance on five-window app-level data when only training and evaluating on sequences with a minimum length of 20 (right bars), for $k \in \{1, 5, 10, 20\}$.}

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Minimum Sequence Length (III)

- *CT* still has the best $HR@k, \forall k \in \{1, 5\}$, and the best $MRR@k, \forall k \in \{1, 5, 10, 20\}$.
- No large changes for *AR*, *SR*, and *SR_BR*.
- Performance of NN-based models improves $\rightarrow$ RNN-based models benefit from being trained and evaluated on longer sequences.
- What if instead we train on all sequences and only evaluate on long sequences?
  - *CT* is again the best performer, but absolute performance is considerably worse.
  - All neural models perform considerably worse than above.
  - This is somehow surprising: the full training dataset is considerably larger than the long-sequences-only dataset.
- We conclude that performance on long sequences benefits from training on long sequences only.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Position in Test Sequence (I)

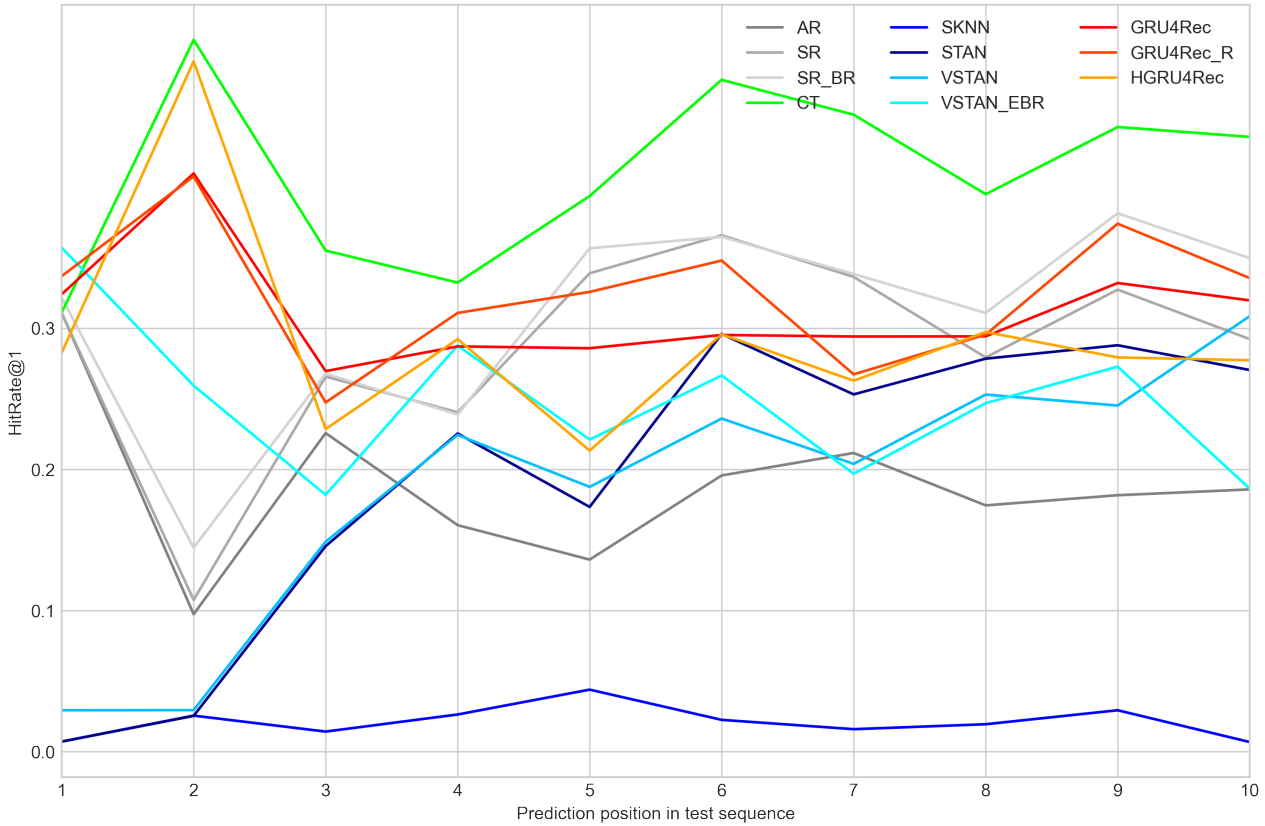

Figure XXX: *HR*@1 performance across the first ten prediction positions on five-window app-level data.

What next? Modeling human behavior using smartphone usage data and (deep) recommender systems

Simon Wiegrebe

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Position in Test Sequence (II)

- *VSTAN_EBR* performs well on early positions but performance deteriorates for longer test sequences.
- For the remaining neighborhood-based models, the behavior is reversed: very weak performance on the first positions but an increase in performance with increasing position.
- *CT* and the three NN-based models do not show any clear trend across positions.
- Shortcomings of this visual analysis:
  - Hard to identify a pattern when the actual question is whether an algorithm performs better on "early" or "late" positions
  - Positions > 10 are not considered at all
- Introduce a cutoff after the 2nd, 5th, or 10th event to be predicted, then compare pre- and post-cutoff performance

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Position in Test Sequence (III)

model	position <= 2	position > 2	position <= 5	position > 5	position <= 10	position > 10
AR	0.2269	0.1892	0.2082	0.1934	0.2045	0.1898
SR	0.2307	0.2768	0.2514	0.2798	0.2676	0.2516
SR_BR	0.2520	0.2854	0.2660	0.2904	0.2838	0.2468
CT	0.3878	0.3818	0.3766	0.4012	0.3911	0.3710
SKNN	0.0142	0.0167	0.0193	0.0111	0.0193	0.0061
STAN	0.0145	0.2298	0.0843	0.2602	0.1268	0.2385
VSTAN	0.0295	0.2230	0.0950	0.2469	0.1270	0.2577
VSTAN_EBR	0.3180	0.2058	0.2807	0.1903	0.2709	0.1405
GRU4Rec	0.3581	0.2984	0.3264	0.3098	0.3208	0.3173
GRU4Rec_R	0.3659	0.2827	0.3342	0.2816	0.3304	0.2311
HGRU4Rec	0.3639	0.2593	0.3132	0.2665	0.3073	0.2542

Table XXX: *HR*@1 performance results on five-window app-level data, by positional cutoff within test sequence.

- Performance for all NN-based models actually worsens for later positions in the test sequences
- They appear to struggle with later positions in the prediction sequences and, consequently, with long prediction sequences - if training is not tailored towards them.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Removing ON and OFF Events (I)

- Short sequence length is a key issue and potential performance bottleneck for app-level analysis.
- By construction, the first and last event of each sequence are ON and OFF events, respectively, and hardly informative
- ON-OFF sequences make up 38.91% of all app-level sequences.
- To increase information density across and within sequences as well as average sequence length, here we analyze the effect of dropping all ON and OFF events from the app-level data.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Removing ON and OFF Events (II)

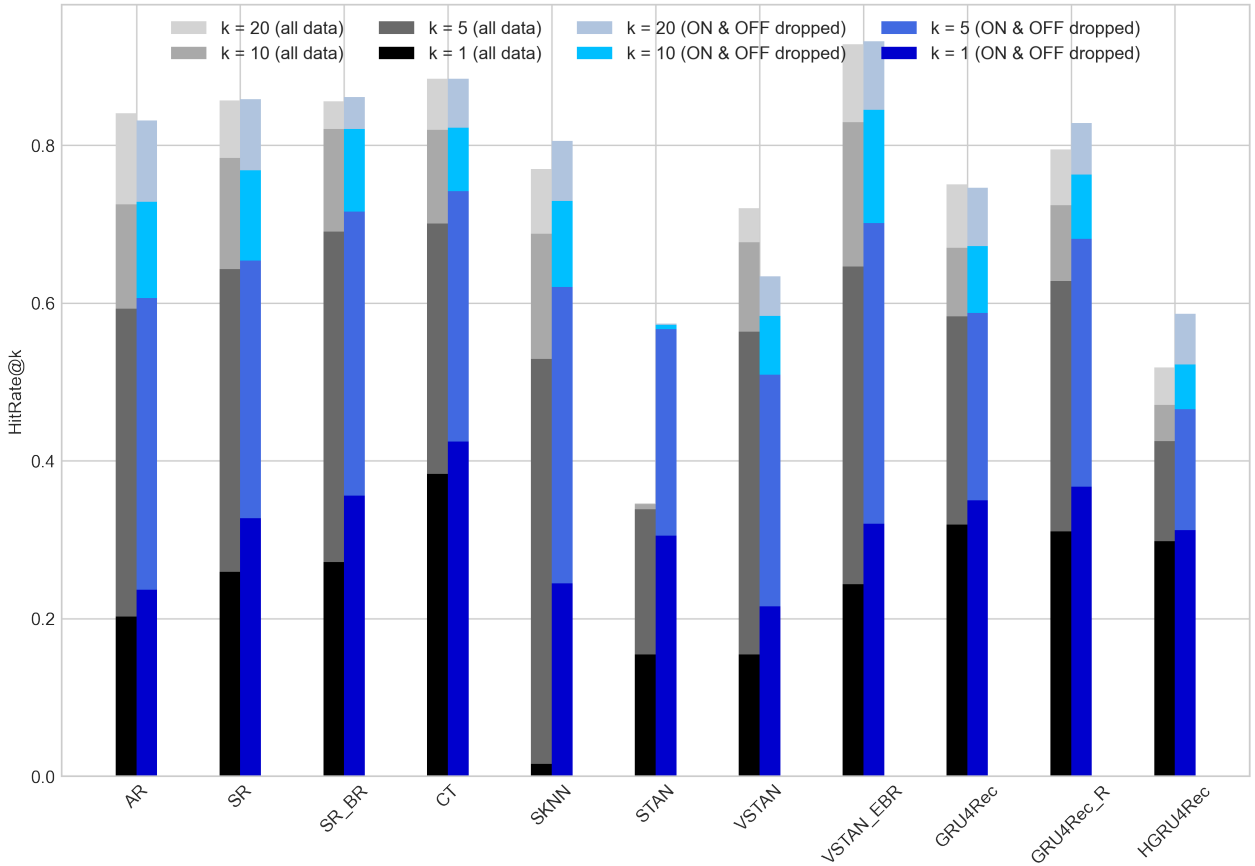

Figure XXX: *HR@k* performance comparison between full five-window app-level data (left bars) and five-window app-level data after dropping all ON and OFF events (right bars), for $k \in \{1, 5, 10, 20\}$.}

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Removing ON and OFF Events (III)

- Dropping ON and OFF events increases performance, especially for the top-ranked recommendation (*HR*@1).
- Substantial performance increase for the nearest neighbor-based models *SKNN* and *STAN*.
  - These models do not have any user-level heuristics.
  - Since the first event of all app-level sequences is a (non-informative) ON event, it is very hard for these models to find similar sequences based on such unspecific information.
- Drawback:
  - Model performance now reflects the models' capability to predict behavioral sequences *conditional* upon the sequence not being an ON-OFF sequence and upon the next event being neither an ON nor an OFF event.
  - In our case, dropping ON and OFF events leads to the exclusion of 49.92% of all app-level sequences.
  - This conditional analysis is hardly representative of overall user behavior anymore.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Category-level Prediction (I)

- Ultimate goal: predict human behavioral sequences → consider next-category prediction instead of next-app prediction.
- For evaluation, we thus simply consider the app category: for instance, "messaging" instead of "WhatsApp".
- If performance improves considerably: our models actually learn more about behavioral sequences than previously thought.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Category-level Prediction (II)

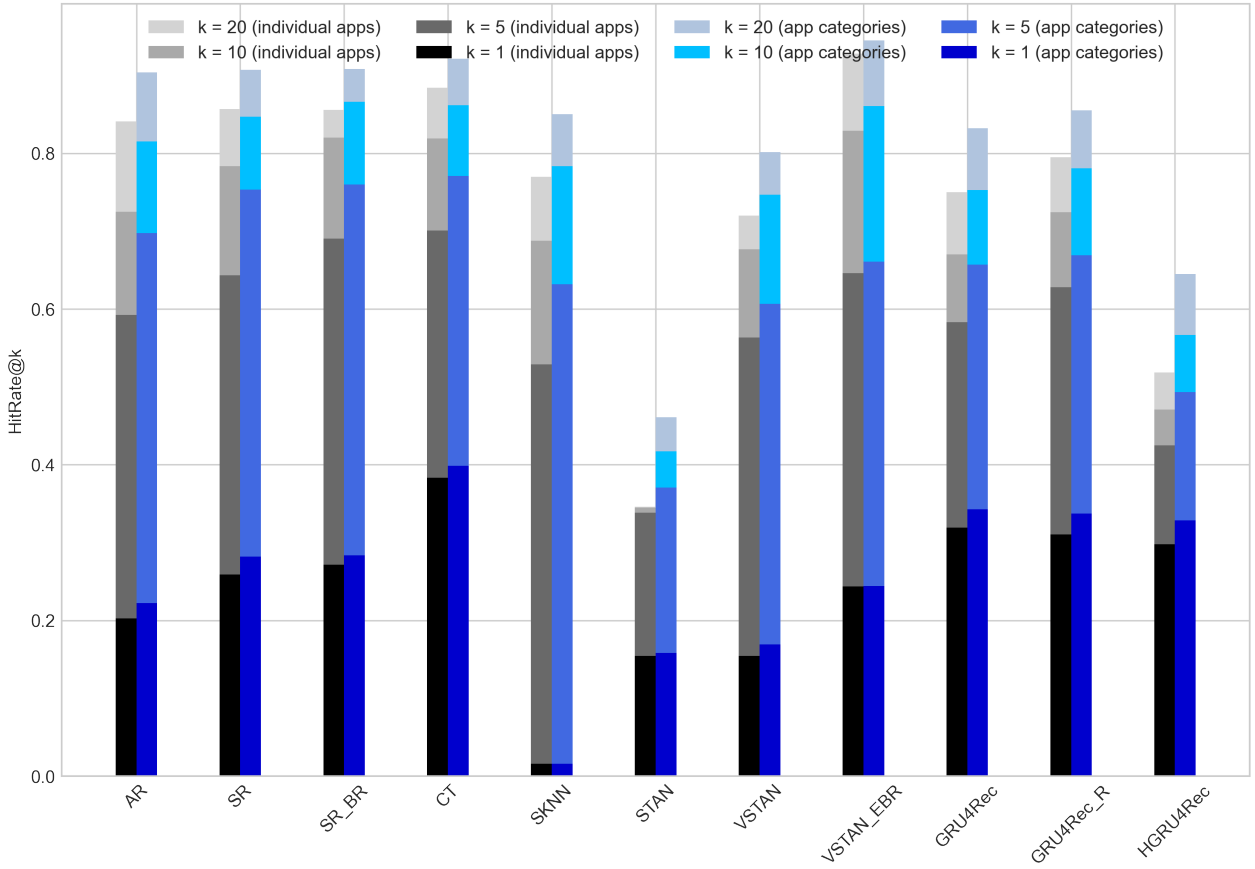

Figure XXX: *HR@k* performance increases on five-window app-level data when only considering app categories for evaluation (left bars), instead of considering the individual apps as well (right bars), for $k \in \{1, 5, 10, 20\}$.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Category-level Prediction (III)

- Performance increases for non-neural methods: small for $k = 1$, noticeable for larger k.
- Performance increases for neural methods: more pronounced, especially for larger k.
- In general: performance increases are proportional to standard app-level performance.
- Conclusions:
  - Next-app prediction performance is largely indicative of how much the algorithms truly learn in terms of behavioral sequence modeling.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Embedding Analysis (I)

- Can deep learning models learn smartphone app semantics?
- Do apps from a common app category form clusters in the embedding space?
- We add an embedding layer ($d = 128$) to the otherwise fully tuned *GRU4Rec* model and extract it after training.
- Then, we apply the t-distributed stochastic neighbor embedding (TSNE) dimensionality reduction technique (Hinton and Roweis 2002) to obtain two-dimensional app embeddings.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Embedding Analysis (II)

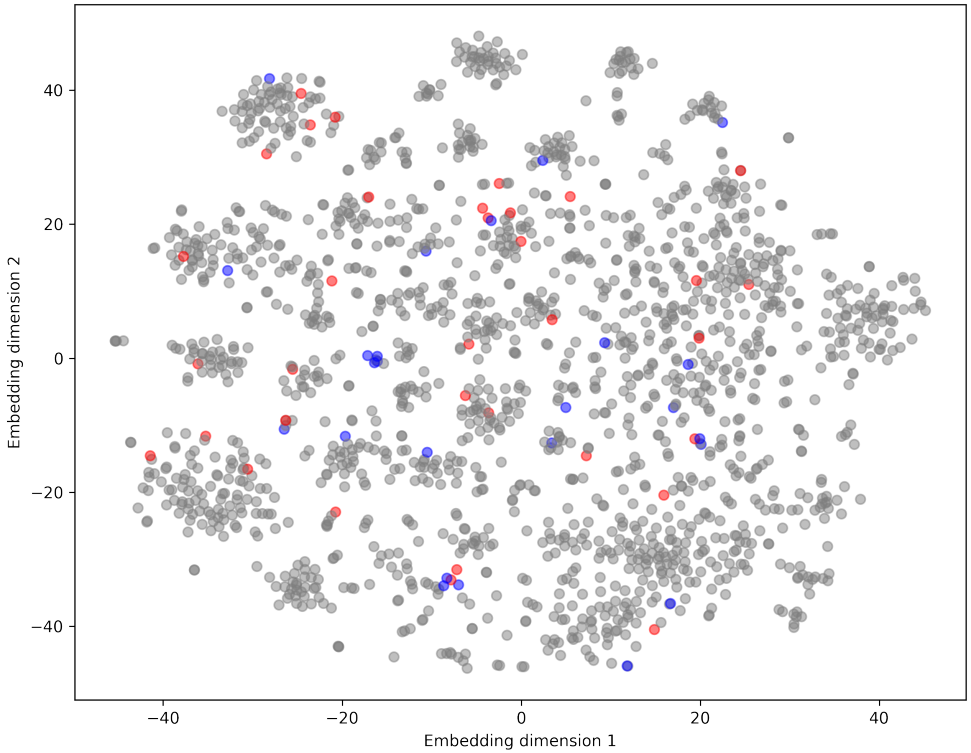

Figure XXX: App category-based clustering of app-level embeddings. Blue dots represent apps categorized as *Messaging*, red dots represent apps categorized as *Social Networks*. For illustration, app embeddings are reduced to a dimensionality of two.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Embedding Analysis (III)

- No apparent category-level clustering is recognizable.
- In numbers: only for 11.67% of apps their most similar app (i.t.o. cosine similarity) is from the same category.
- App embeddings learned by the *GRU4Rec* model are not very successful at learning true app-level semantics.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Embedding Analysis (IV)

- Alternatively, we might want to start off with a data-driven clustering approach, based on positioning within the embedding space.
- We use k-means clustering on the untransformed 128-dimensional app embeddings ($k = 15$).
- Then we look at potential accumulations of app categories within each one of the 15 clusters.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Embedding Analysis (V)

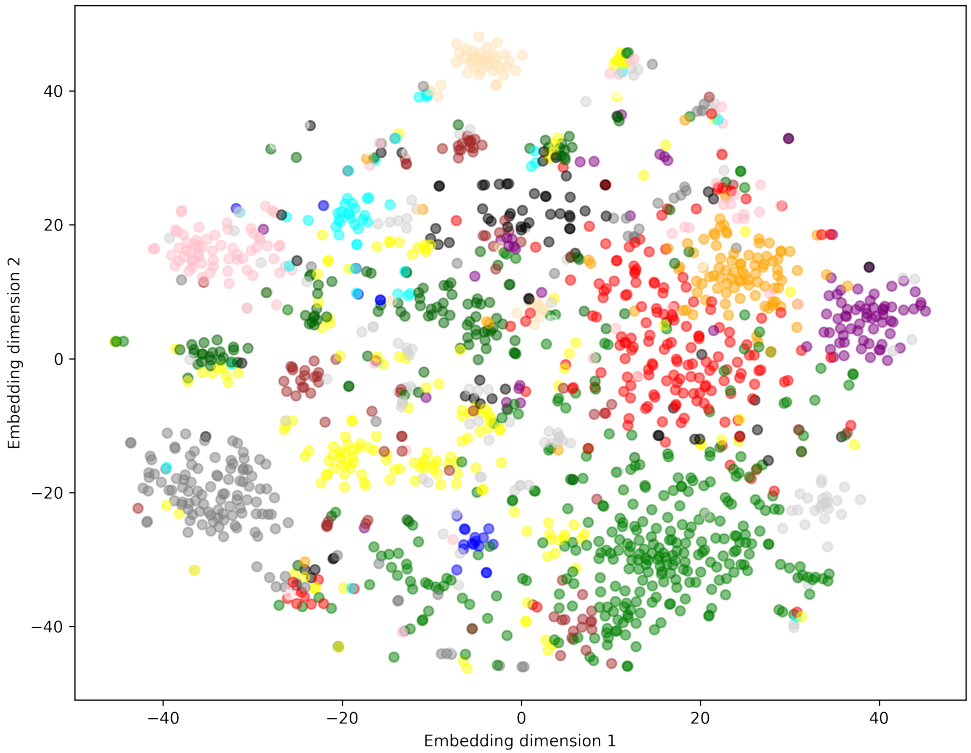

Figure XXX: k-means clustering of app-level embeddings ($k = 15$). For illustration, app embeddings are reduced to a dimensionality of two.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Embedding Analysis (VI)

- Moccasin-colored cluster (center top): high prevalence of apps from only few but related categories:
  - 32 out of a total 52 apps (i.e., more than 60%) are either camera or image editing apps.
- However, the vast majority of clusters are very much dispersed across the app space, with few to no intra-cluster app category clustering.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

App-level Results

Embedding Analysis (VII)

- Finally, we briefly consider app analogies.
- We experimentally construct app analogies such as "Messaging 1 + Social Network 1 - Social Networks 2 = ???".
- We find no meaningful app analogies in our embeddings:
  - App analogies are conceptually much less intuitive than word embeddings.
  - The overall quality of our *GRU4Rec* embeddings is rather low, just like the degree of clustering amongst apps from the same category.
- Altogether:
  - Next-app prediction is possible and informative given our app-level data $\rightarrow$ the algorithms do learn behavioral patterns based on app usage data.
  - This learning is not profound enough as for app semantics to be recognizable.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

Sequence-level Results

Overall Performance (I)

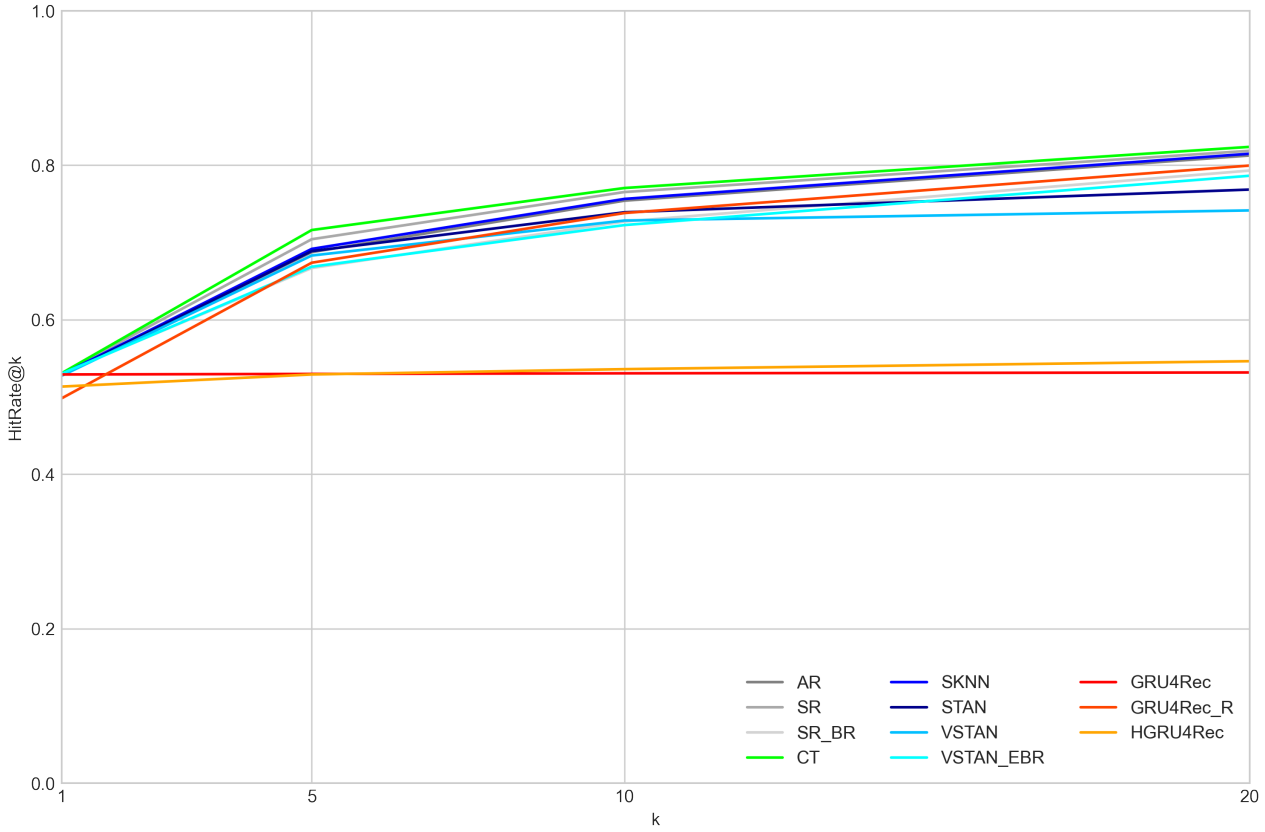

Figure XXX: *HR@k* performance for $k = 1, 5, 10,$ and 20 on five-window sequence-level data.

What next? Modeling human behavior using smartphone usage data and (deep) recommender systems

Simon Wiegrebe

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

Sequence-level Results

Overall Performance (II)

- All algorithms show a strong performance in terms of *HR*@1, at around 50%.
- Performance increases with increasing k are much lower than in the app-level setting $\rightarrow$ *HR*@20 performance worse than in the app-level case for 7 algorithms.
- As in the app-level setting: *CT* is the best performer.
- *GRU4Rec* and *HGRU4Rec* are the weakest performers for all metrics where $k \geq 5$.
  - Surprising since sequence-level sequences are mostly rather long
- *CT* covers 95% of all tokens, while *GRU4Rec* only covers 8%.
- Popularity bias is rather low (between 5% and 8%) for all models.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

Sequence-level Results

Removing ON-OFF Tokens (I)

- The high *HR*@1 performance across all algorithms is suspicious.
- It might simply be the consequence of the high prevalence of ON-OFF tokens (51.06%).
- Removing all ON and OFF events from the underlying app-level data translates into removing all ON-OFF tokens from the sequence-level data.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

Sequence-level Results

Removing ON-OFF Tokens (II)

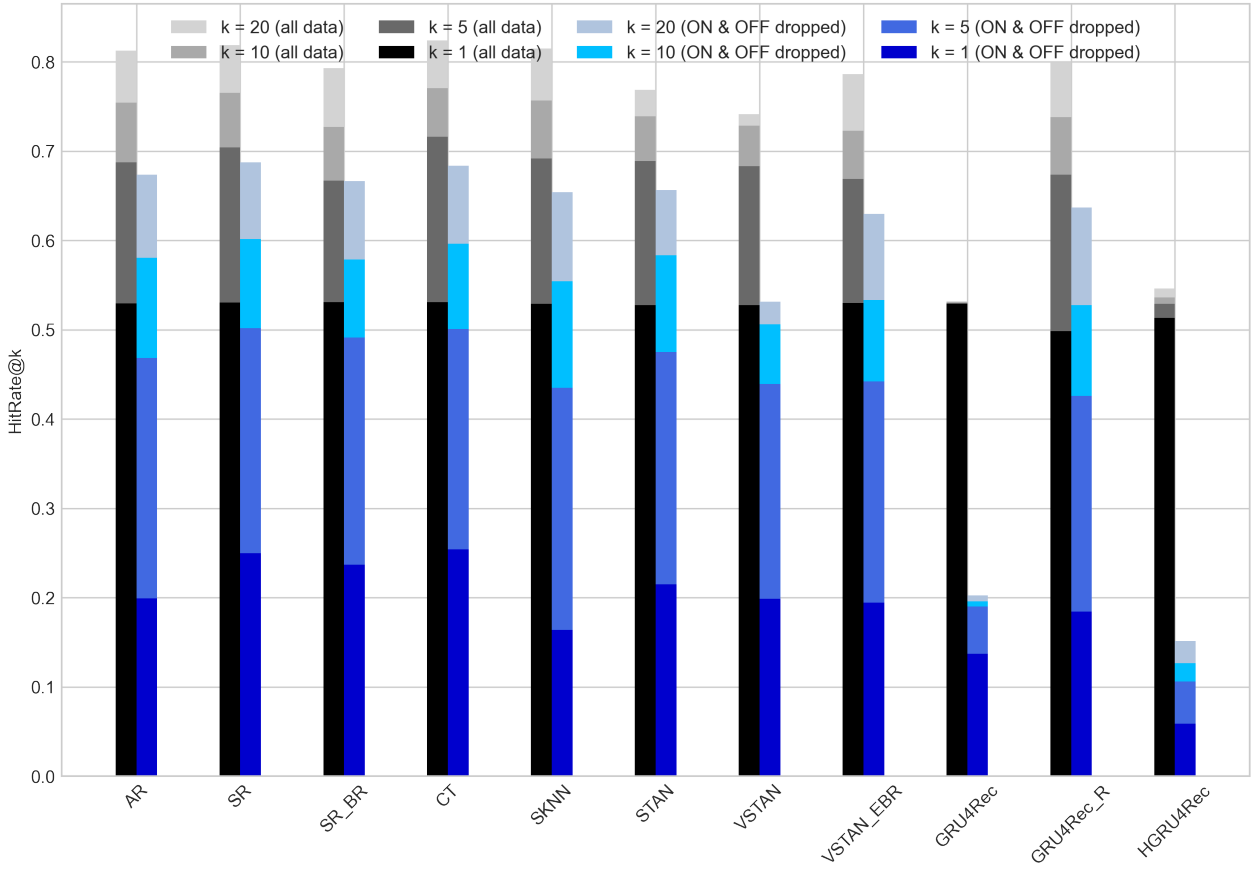

Figure XXX: *HR@k* performance comparison for all selected algorithms between full five-window sequence-level data (left bars) and five-window sequence-level data after dropping all ON and OFF events (right bars), for $k \in \{1, 5, 10, 20\}$.

Sequence-level Results

Removing ON-OFF Tokens (III)

- Performance drops for all algorithms after excluding ON-OFF tokens, especially i.t.o. *HR*@1.
- Average performance drop (across all algorithms): 63.88%.
- *CT* is still the best performer, *GRU4Rec* and *HGRU4Rec* are the worst performers for all metrics.
- On the unrestricted data, 10 out of 11 algorithms predict an ON-OFF token as top ranked recommendation in more than 95% of all cases.
  - As a consequence, *HR*@1 on non-ON-OFF events is close to 0 for all algorithms.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

Sequence-level Results

Position in Test Sequence (I)

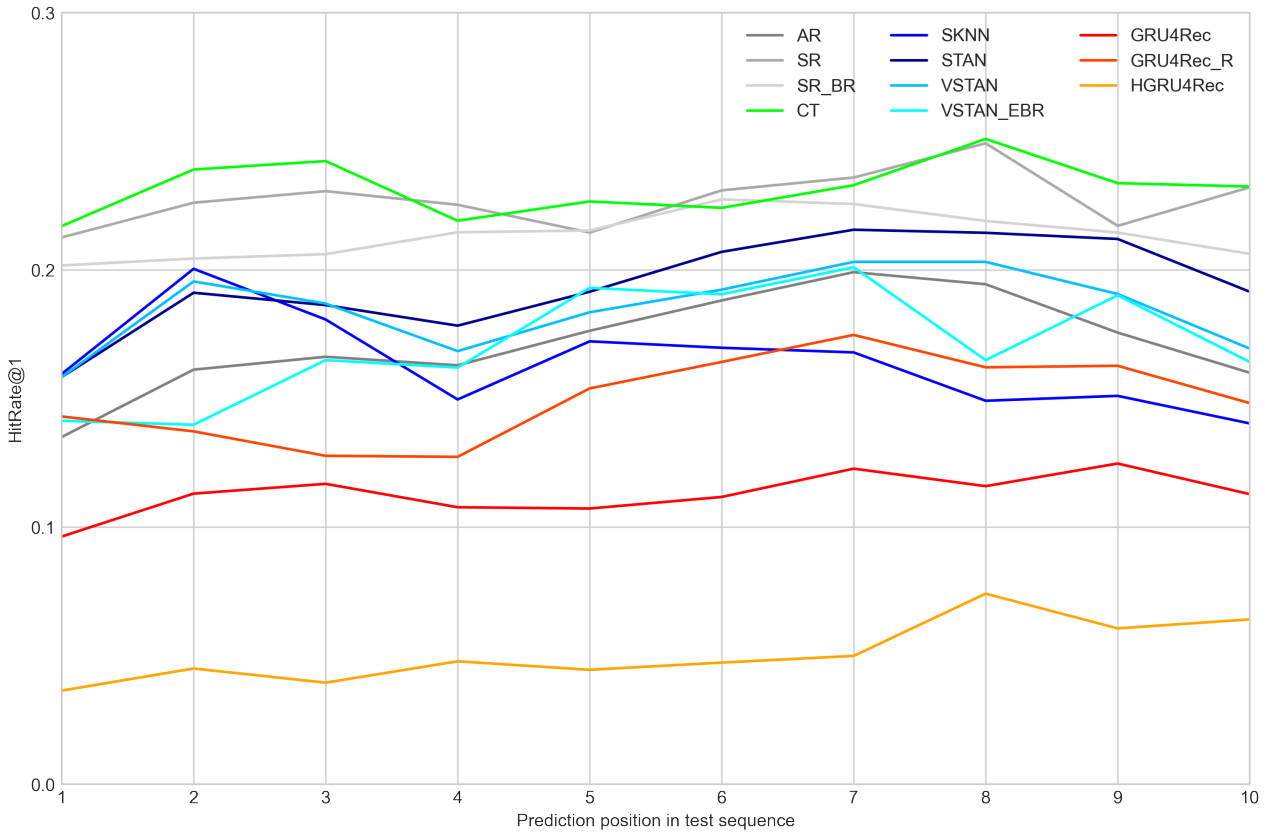

Figure XXX: *HR*@1 performance across the first ten prediction positions on five-window sequence-level data for all selected algorithms.

Sequence-level Results

Position in Test Sequence (II)

- We use sequence-level data with all ON and OFF events removed from the underlying app-level data.
- We do not see a clear trend for any of the models as the position of the event within the test sequence increases.
- Performance for the first position is relatively weak for all algorithms.
- Again, we introduce a cutoff after the 2nd, 5th, or 10th event to be predicted and compare pre- and post-cutoff performance

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

Sequence-level Results

Position in Test Sequence (III)

model	position <= 2	position > 2	position <= 5	position > 5	position <= 10	position > 10
AR	0.1478	0.2024	0.1598	0.2057	0.1711	0.2096
SR	0.2193	0.2515	0.2218	0.2543	0.2270	0.2581
SR_BR	0.2030	0.2387	0.2083	0.2413	0.2132	0.2454
CT	0.2278	0.2556	0.2288	0.2583	0.2314	0.2622
SKNN	0.1798	0.1629	0.1725	0.1625	0.1645	0.1635
STAN	0.1746	0.2174	0.1808	0.2206	0.1936	0.2226
VSTAN	0.1768	0.1999	0.1783	0.2019	0.1846	0.2037
VSTAN_EBR	0.1406	0.1972	0.1595	0.1991	0.1704	0.2019
GRU4Rec	0.1045	0.1389	0.1081	0.1416	0.1127	0.1462
GRU4Rec_R	0.1402	0.1870	0.1377	0.1916	0.1494	0.1963
HGRU4Rec	0.0405	0.0597	0.0426	0.0614	0.0504	0.0621

Table XXX: *HR*@1 performance results on five-window sequence-level data, by positional cutoff within test sequence.

- All models except *SKNN* perform better on later positions of the test sequences
- The precise positioning of the cutoff is not decisive.

Sequence-level Results

Position in Test Sequence (IV)

- For NN-based models, this performance improvement for later events is in line with what we expected.
- Difference in performance across positions for app- versus sequence-level data:
  - App-level setting: predominantly short sequences.
  - Sequence-level setting: mostly long sequences.
- This corroborates our previous conclusion: differences in sequence lengths between training and evaluation data negatively affect the performance of NN-based algorithms.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

6 References

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References

Hinton, Geoffrey, and Sam T Roweis. 2002. "Stochastic Neighbor Embedding." In *NIPS*, 15:833–40. Citeseer.

1 Introduction

2 Theoretical Framework

3 Data

4 Methodology

5 Results

6 References