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

Master Thesis Presentation

Simon Wiegrebe

Supervisor: Dr. David Rügamer

October 01, 2021

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

Simon Wiegrebe

1 Motivation

2 Data

3 Modeling

4 Evaluation

5 App-level Results

6 Sequence-level Results

7 Discussion

8 References

1 Motivation

Introduction

- Smartphone usage has been becoming a valuable source of data in recent years:
  - large volume
  - ubiquitous
  - easily accessible
  - clean
  - representative of actual human behavior
- Behavioral researchers: investigating human behavioral traits through smartphone usage
- Most behavioral research: association between smartphone usage patterns and pre-established personality traits
- Here: data-centric approach to the modeling of human behavioral sequences

Research Idea

- Smartphone usage data from a PhoneStudy project (Stachl et al. 2019)
- There is a natural sequential order in the data:
  - An app session starts by switching on the screen and ends by switching it off
  - The apps used in between, ordered by their timestamps, as well as the ON and OFF tokens form the events of an app session
- Model behavioral sequences by means of next-event prediction
- Large number of possible events + sequential data $\rightarrow$ Use sequence-aware recommender system (RS) algorithms

2 Data

Description

- PhoneStudy dataset from a mobile sensing research project (Stachl et al. 2019)
- 310 users, study period from October 29, 2017 through January 22, 2018
- Each app usage assigned exact opening date and time

userID	timestamp	sessionID	appID
1	1.511423e+09	1	1392
1	1.511423e+09	1	1389
1	1.511424e+09	2	1392
1	1.511424e+09	2	1389
1	1.511424e+09	3	1392
...	...	...	...
310	1.515952e+09	844295	1389
310	1.515953e+09	844296	1392
310	1.515953e+09	844296	1389

Table XXX: Excerpt of anonymized app-level data.

App-level Representation

- In language modeling:
  - Tokens $\widehat{=}$ words
  - Sentence $\widehat{=}$ concatenation of tokens ending with a period
- Here:
  - Tokens $\widehat{=}$ apps
  - Sentences $\widehat{=}$ sessions
- Objective: next-app prediction
  - Predicting the next app a user is going to use in a given session
- Mostly very short sessions

Sequence-level Representation

- How to address the issue of short session length?
- Focus on behavior, not individual apps
  - App-level sessions := concatenations of app-level *categories*
  - These categories were pre-established by Stachl et al. (2020)
    - E.g.: "WhatsApp" $\rightarrow$ "Messaging"
- Now:
  - Tokens $\widehat{=}$ app-level sessions
  - Sentences $\widehat{=}$ daily concatenations of a user's sessions
- For the sake of unambiguousness: use the terms "sequence" and "event"

Summary Statistics

	App-level	Sequence-level
Number of events	4,314,830	720,379
Number of unique events	2,488	2,454
Number of sequences	844,296	9,377
Number of users	310	310
Sequences per user	2,723.54	30.25
Mean number of events per sequence	5.11	76.82
1st quartile of number of events per sequence	2.0	34.0
Median number of events per sequence	4.0	66.0
3rd quartile of number of events per sequence	6.0	106.0

Table XXX: Summary statistics of app-level and sequence-level data.

- Drawback of sequence-level analysis: data size
  - events in sequence-level $\approx$ sequences in app-level

3 Modeling

Definitions and Terminology

- Baseline model := non-NN-based model
- Session-based model: no incorporation of user-level information (user ID)
- Session-aware model: incorporation of user-level information
- $s = (s_1, s_2, \ldots, s_m)$: sequence of chronologically ordered events
  - s_s: last "known" event in the sequence
  - s_{s+1}: event we seek to predict
- i: candidate event for s_{s+1}

Session-based Baseline Models (I)

- *AR* and *SR* (Ludewig and Jannach 2018)
  - are based on co-occurrence frequencies
  - only take into account s_s when making a prediction
- *AR*
  - simply counts co-occurrences of s_s with *any* i
  - normalizes this count by the number of all co-occurrences
- *SR*
  - accounts for sequential event order
  - only counts co-occurrences where s_s precedes any i
  - decreases the weight if other events occurred in-between

Session-based Baseline Models (II)

- The neighborhood-based *SKNN* (Jannach and Ludewig 2017)
  - defines a neighborhood of most similar past sequences
  - determines similarity between s and neighbor sequences
  - computes score as sum of similarity scores across all sequences which contain i
- *STAN* (Garg et al. 2019) and *VSTAN* (Ludewig et al. 2021) extend *SKNN*, for instance by
  - accounting for event recency in s using decay functions
  - accounting for sequence recency of neighbor sequences

Session-based Neural Models

- *GRU4Rec* (Hidasi et al. 2015)
  - initially one-hot encodes single input events
  - feeds input vectors into a Gated Recurrent Unit (GRU) layer
  - uses pairwise ranking losses for training
  - outputs, for each event, the likelihood of being next in the sequence

Session-aware Neural Models

- *HGRU4Rec* (Quadrana et al. 2017)
  - is a user-aware extension of *GRU4Rec*
  - contains a short- and long-term memory GRU layer
  - generates recommendations for each event in a sequence through a session-level GRU (like *GRU4Rec*)
  - updates an additional user-level GRU at the end of each sequence
  - employs its hidden state for initialization of the session-level GRU at the beginning of the next sequence

Extensions

- We use (a combination of) 3 different heuristics for some session-based algorithms
- Extensions contribute user-level information from past sequences $\rightarrow$ session-awareness

1) *Extend* prepends events from the user's preceding sequence if s is short

2) *Boost* increases the score of i if i has occurred in the user's past sequences

3) *Remind* adds a reminder score to the original model score

Implementation

- Implementation of models and extensions based on Latifi, Mauro, and Jannach (2021)[1]
- We perform all modeling, evaluation, and analysis tasks in Python

[1] https://github.com/rn5l/session-rec/

4 Evaluation

Train-Validation-Test Split

- Time-ordered and user-clustered data
- Standard time-agnostic cross-validation not applicable
- Last-event split method, applied twice:
  1) Clip off each user's last sequence $\rightarrow$ test set
  2) Clip off each user's last sequence from the remaining data $\rightarrow$ validation set
- Each user required to have ≥ 3 sequences
- Additionally: split study period into 5 equally long sub-periods (windows)
  - Apply train-validation-test split to all 5 windows
  - Average performance results across all 5 test sets

Evaluation Protocol

- Evaluate predictions for all but the first test sequence events
- Preferable to *last-item prediction*
- How to define ground truth:
  - Mostly interested in predicting the single next action of a user
  - Our definition: only the event observed at specific position

Evaluation Metrics (I)

- Target variable follows a multinomial distribution with large number of categories
- We wish to quantify the goodness of our recommendation list of length k
- We wish to perform next-event prediction with our ground truth being a single event
- Let n be the total number of events to be predicted

Evaluation Metrics (II)

1) Hit Rate (HR): $HR@k$ is simply the fraction of events for which the corresponding recommendation list of length k, $rl(k)_i$, includes the ground truth, y_i:

$$HR@k = \frac{\sum_{i=1}^{n} \mathbb{1}_{rl(k)_i}(y_i)}{n}$$

2) Mean Reciprocal Rank (MRR): $MRR@k$ additionally accounts for the ranking within the recommendation list. $MRR@k$ computes the reciprocal rank of the ground truth within the recommendation list, rr_i, then averages this reciprocal rank across all n events:

$$MRR@k = \frac{\sum_{i=1}^{n} rr_i}{n}$$

- We consider $HR@k$ and $MRR@k$ for $k \in \{1, 5, 10, 20\}$

Tuning

- Simple random search with budget 100 for each algorithm
- Hyperparameter search spaces as in Latifi, Mauro, and Jannach (2021)
- Tuning on five-window data, then averaging performance to determine optimal hyperparameter configuration
- Tuning metric: $HR@1$

5 App-level Results

Overall Performance

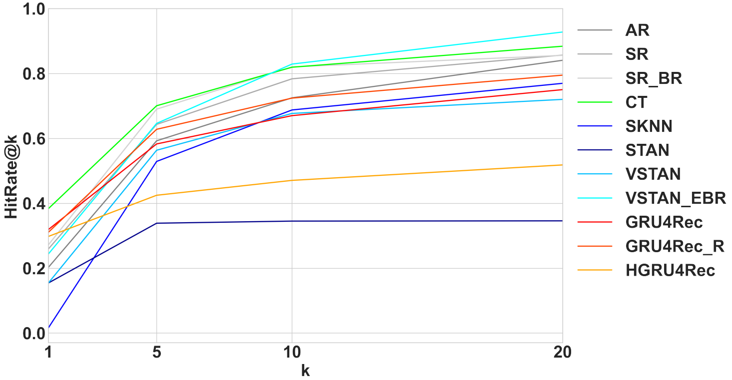

Figure XXX: *HR@k* performance for $k = 1, 5, 10,$ and 20 on five-window app-level data.

- Best performer i.t.o. *HR*@1 and *HR*@5: *CT*
- Best performer i.t.o. *HR*@10 and *HR*@20: *VSTAN_EBR*
- Strong *HR*@1 performance of NN-based models

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 → RNN-based methods do not have "much to learn from"
- Hypotheses:
  - Better performance of NN-based models on longer sequences
  - No impact of sequence length on performance of *AR*, *SR*, and *SR_BR*

→ Train and evaluate our models on a subset containing only sequences with at least 20 events.

Minimum Sequence Length (II)

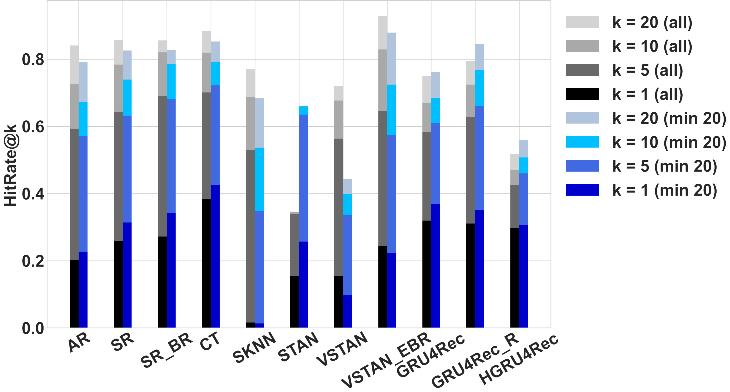

Figure XXX: *HR@k* comparison between performance on full five-window app-level data (left bars) and performance on five-window app-level data when only training and evaluating on sequences with a minimum length of 20 (right bars), for $k \in \{1, 5, 10, 20\}$.}

- *CT* still best performer for *HR@1* and *HR@5*
- No large changes for *AR*, *SR*, and *SR_BR*
- Performance of NN-based models improves

Minimum Sequence Length (III)

- What if instead we train on all sequences and only evaluate on long sequences?
  - *CT* still best performer
  - All neural models perform considerably worse
  - Surprising because the full training dataset is considerably larger
- Conclusion: performance on long sequences benefits from training on long sequences only

Position in Test Sequence (I)

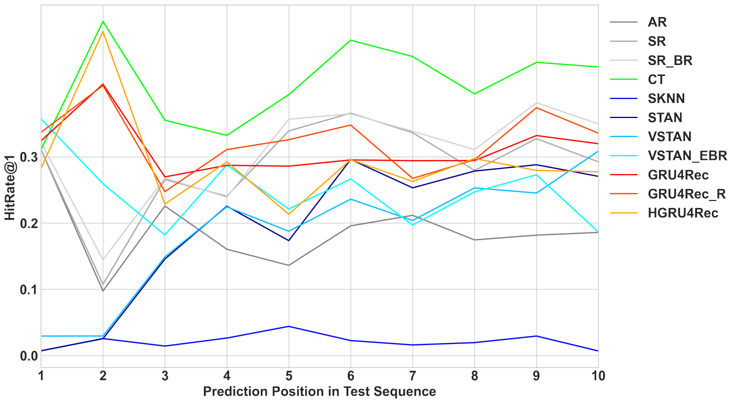

Figure XXX: *HR*@1 performance across the first ten prediction positions on five-window app-level data.

- Initial performance boost for *VSTAN_EBR*
- No clear trend for all other models

Position in Test Sequence (II)

Algorithm	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.

- Worse performance for NN-based models on later positions
- if training is not tailored towards them: NN-based models struggle with later positions in the prediction sequences and, consequently, with long prediction sequences

Removing ON and OFF Events (I)

- Key issue and potential performance bottleneck: short sequence length
- ON and OFF events are hardly informative
- ON-OFF sequences make up 38.91% of all app-level sequences
- Effect of dropping all ON and OFF events from the app-level data?

Removing ON and OFF Events (II)

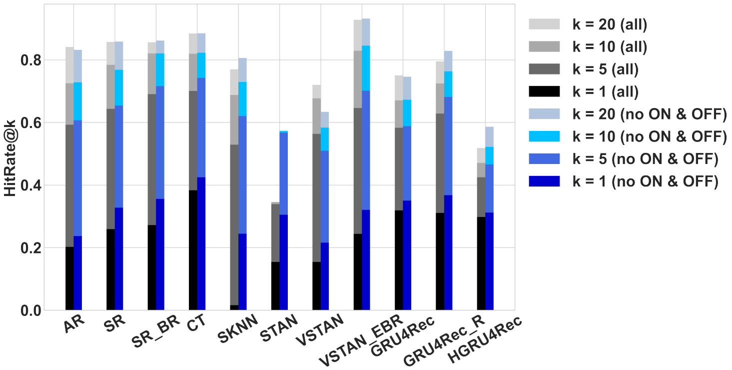

Figure XXX: *HR@k* performance comparison between full five-window app-level data (left bars) and five-window app-level data after dropping all ON and OFF events (right bars), for $k \in \{1, 5, 10, 20\}$.}

- Improvements i.t.o. *HR*@1 across the board
- Substantial improvements for neighborhood-based models
- Drawback: limited representativeness of results

Category-level Prediction (I)

- Ultimate goal: predict human behavioral sequences → consider next-category prediction instead of next-app prediction.
- For evaluation, simply consider app category: e.g., "messaging" instead of "WhatsApp".
- If performance improves considerably: models learn more about behavioral sequences than previously thought

Category-level Prediction (II)

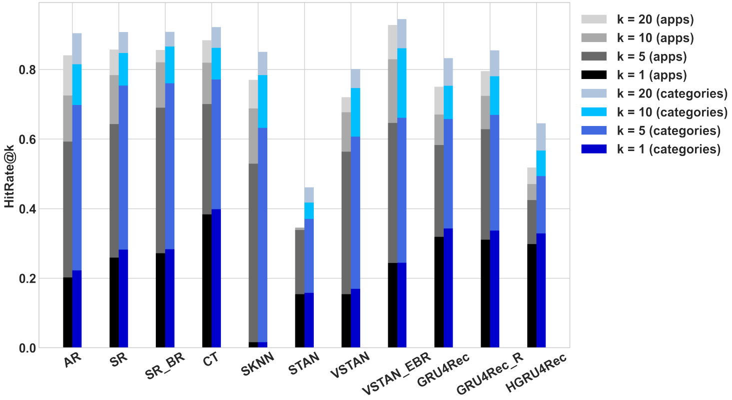

Figure XXX: *HR@k* performance increases on five-window app-level data when only considering app categories for evaluation (left bars), instead of considering the individual apps as well (right bars), for $k \in \{1, 5, 10, 20\}$.

- Performance increases especially for larger k, more pronounced for NN-based methods, and proportional to app-level performance

Embedding Analysis (I)

- Can deep learning models learn smartphone app semantics?
- Do apps from a common app category form clusters in the embedding space? $\rightarrow$ Add an embedding layer ($d = 128$) to *GRU4Rec*
- Apply TSNE (Hinton and Roweis 2002) to obtain two-dimensional app embeddings

Embedding Analysis (II)

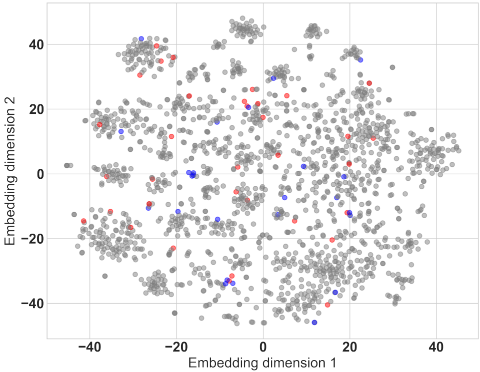

Figure XXX: App category-based clustering of app-level embeddings. Blue dots represent apps categorized as *Messaging*, red dots represent apps categorized as *Social Networks*. For illustration, app embeddings are reduced to a dimensionality of two.

- No category-level clustering recognizable
- Only for 11.67% of apps their most similar app (i.t.o. cosine similarity) is from the same category

Embedding Analysis (III)

- Alternatively: start off with data-driven clustering approach k-means
- Look at potential accumulations of app categories within each cluster

Embedding Analysis (IV)

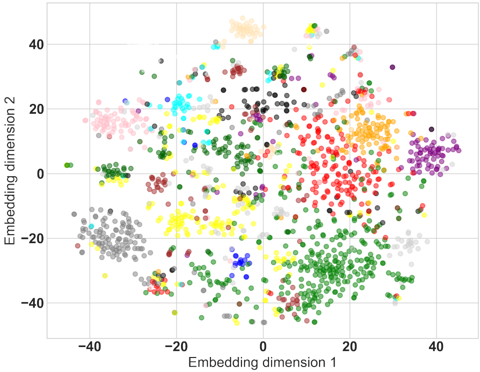

Figure XXX: k-means clustering of app-level embeddings ($k = 15$). For illustration, app embeddings are reduced to a dimensionality of two.

- Moccasin-colored cluster: 32 out of 52 apps (>60%) are camera or image editing apps
- However: vast majority of clusters dispersed across app space, with little intra-cluster app category clustering.

Embedding Analysis (V)

- 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 conceptually much less intuitive than word embeddings
  - Low overall quality of *GRU4Rec* embeddings

6 Sequence-level Results

Overall Performance

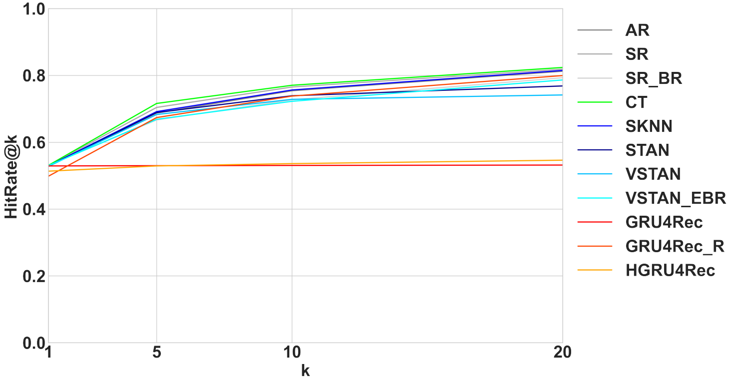

Figure XXX: *HR@k* performance for $k = 1, 5, 10,$ and 20 on five-window sequence-level data.

- Strong *HR*@1 performance by all algorithms
- Low performance increases with increasing k
- *GRU4Rec* and *HGRU4Rec* weakest performers for $k > 1$

Removing ON-OFF Tokens (I)

- Suspiciously high *HR*@1 performance across all algorithms
- High prevalence of ON-OFF tokens (51.06%)
- All algorithms predict ON-OFF tokens (almost) everywhere
  - Predictive performance on other tokens ~0%
- Effect of removing ON and OFF events from underlying app-level data?

Removing ON-OFF Tokens (II)

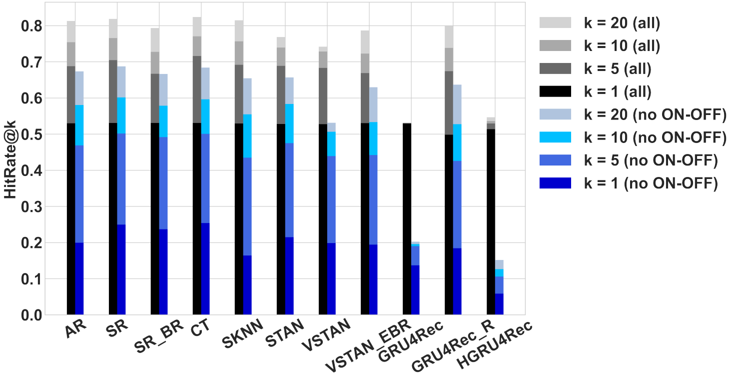

Figure XXX: *HR@k* performance comparison for all selected algorithms between full five-window sequence-level data (left bars) and five-window sequence-level data after dropping all ON and OFF events (right bars), for $k \in \{1, 5, 10, 20\}$.

- Performance drops for all algorithms, especially i.t.o. *HR@1*
- *CT* best, *GRU4Rec* and *HGRU4Rec* worst performers

Position in Test Sequence (I)

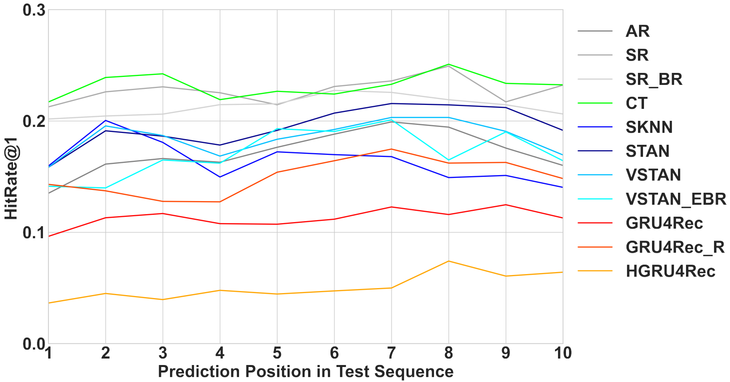

Figure XXX: *HR*@1 performance across the first ten prediction positions on five-window sequence-level data for all selected algorithms.

- ON and OFF events removed from the underlying app-level data
- No clear trend for any of the models

Position in Test Sequence (II)

Algorithm	position <= 2	position > 2	position <= 5	position > 5	position <= 10	position > 10
AR	0.4770	0.5310	0.4839	0.5328	0.4933	0.5351
SR	0.4760	0.5321	0.4833	0.5340	0.4944	0.5361
SR_BR	0.4767	0.5323	0.4856	0.5340	0.4964	0.5360
CT	0.4760	0.5323	0.4857	0.5340	0.4952	0.5362
SKNN	0.4509	0.5312	0.4743	0.5330	0.4900	0.5350
STAN	0.3819	0.5321	0.4351	0.5348	0.4726	0.5364
VSTAN	0.3823	0.5318	0.4345	0.5346	0.4718	0.5363
VSTAN_EBR	0.4776	0.5316	0.4861	0.5334	0.4952	0.5355
GRU4Rec	0.4777	0.5306	0.4838	0.5324	0.4931	0.5347
GRU4Rec_R	0.4506	0.4998	0.4574	0.5015	0.4712	0.5029
HGRU4Rec	0.3840	0.5172	0.4257	0.5200	0.4519	0.5231

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 not very relevant

Position in Test Sequence (III)

- For NN-based models: performance improvement for later events in line with expectations
- Comparison app- versus sequence-level data:
  - App-level setting: predominantly short sequences
  - Sequence-level setting: mostly long sequences
- Corroborates our previous conclusion: differences in sequence lengths between training and evaluation data negatively affect the performance of NN-based algorithms.

7 Discussion

Conclusion (I)

- By and large, strong predictive performance of most algorithms
- NN-based models mostly perform well i.t.o. *HR*@1 and *HR*@5
  - Amongst them, *HGRU4Rec* is often the weakest one
- NN-based model performance is prone to sequence length and data size
- NN-based models are very expensive i.t.o. runtime and computational effort
- Simple, non-NN models are the preferable modeling choice for our data

Conclusion (II)

- *CT* recommendable i.t.o. *HR*@1 and *HR*@5, no tuning
- *SR* exhibits strong performance i.t.o. *HR*@10 and *HR*@20, fast
- No overarching user-level effects in our data
  - For predicting future behavioral sequences of a particular user, not overly helpful to know this particular person's past smartphone usage patterns
- User-level extensions mostly effective, especially for short sequences and early positions
  - not due to some profound user-level learning
  - instead, addressing technical weaknesses of the session-based baseline algorithm
  - e.g., *VSTAN_EBR* alleviates poor early-position performance of other neighborhood-based models stemming from low informational content in short sequences

Limitations

- Dataset size: potentially giving a relative advantage to non-neural methods
- Algorithm selection: not including some of the modern sophisticated approaches, e.g., *BERT4Rec* (Sun et al. 2019)
    - Attention-based models require even more training data
    - Their main advantage is the better handling of *long-term* dependencies while we mostly have *short* sequences

Suggestions for Future Research

- Increased dataset size: new PhoneStudy dataset → Investigate impact of data size on (NN-based) model performance
- Information extraction: incorporation of duration, exact daytime, and geolocation of app usage
- Transfer learning: use of pre-trained transformers?

8 References

References I

Garg, Diksha, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2019. "Sequence and Time Aware Neighborhood for Session-Based Recommendations: Stan." In *Proceedings of the 42nd International Acm Sigir Conference on Research and Development in Information Retrieval*, 1069–72.

Hidasi, Balázs, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. "Session-Based Recommendations with Recurrent Neural Networks." *arXiv Preprint arXiv:1511.06939*.

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

References II

Jannach, Dietmar, and Malte Ludewig. 2017. "When Recurrent Neural Networks Meet the Neighborhood for Session-Based Recommendation." In *Proceedings of the Eleventh Acm Conference on Recommender Systems*, 306–10.

Latifi, Sara, Noemi Mauro, and Dietmar Jannach. 2021. "Session-Aware Recommendation: A Surprising Quest for the State-of-the-Art." *Information Sciences* 573: 291–315.

Ludewig, Malte, and Dietmar Jannach. 2018. "Evaluation of Session-Based Recommendation Algorithms." *User Modeling and User-Adapted Interaction* 28 (4-5): 331–90.

Ludewig, Malte, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2021. "Empirical Analysis of Session-Based Recommendation Algorithms." *User Modeling and User-Adapted Interaction* 31 (1): 149–81.

References III

Quadrana, Massimo, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. "Personalizing Session-Based Recommendations with Hierarchical Recurrent Neural Networks." In *Proceedings of the Eleventh Acm Conference on Recommender Systems*, 130–37.

Stachl, Clemens, Quay Au, Ramona Schoedel, Samuel D Gosling, Gabriella M Harari, Daniel Buschek, Sarah Theres Völkel, et al. 2020. "Predicting Personality from Patterns of Behavior Collected with Smartphones." *Proceedings of the National Academy of Sciences* 117 (30): 17680–7.

Stachl, Clemens, Ramona Schoedel, Quay Au, Sarah Völkel, Daniel Buschek, Heinrich Hussmann, Bernd Bischl, and Markus Bühner. 2019. "The Phonestudy Project." OSF. https://doi.org/10.17605/OSF.IO/UT42Y.

References IV

Sun, Fei, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. "BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer." In *Proceedings of the 28th Acm International Conference on Information and Knowledge Management*, 1441–50.