Behaviour-Based Working Memory Capacity Classification Using Recurrent Neural Networks

Mazen Salous and Felix Putze *

University of Bremen - Cognitive Systems Lab
Bremen - Germany

Abstract. A user's working memory capacity is a crucial factor for successful Human Computer Interaction (HCI). While reliable tests for working memory capacity are available, they are time-consuming, stressful, and not well-integrated into HCI applications. This paper presents a classifier based on Long Short Term Memory networks to exploit sparse temporal dependencies in behavioural data, collected in a complex, memory-intense interaction task, to classify working memory capacity. A cognitive user simulation is introduced to generate additional training data episodes that follow the behaviour of existing real data. We show that the classifier outperforms a linear baseline especially for short segments of data.

1 Introduction and Related Work

Humans employ several perceptual, cognitive, and motor capabilities while interacting with computers. However, a key to successful Human Computer Interaction (HCI) is working memory (WM) capacity, which is strongly correlated to general intelligence [1]. WM is relevant to preserve attention and control all cognitive interactions. Cognitive Load Theory [2] suggests that overloading a user's WM leads to deteriorated performance and requires system adaptation.
Due to its high importance in HCI contexts, there exist, on the one hand, many approaches e.g. [3] aiming at dynamically measuring working memory load by sensor data. However, such approaches do not estimate general, person-specific WM capacity. On the other hand, there exist also standard approaches for measuring WM capacity by implementing widely used WM capacity measures (e.g., Operation: Memory-Update (MU) and Reading span [4]). For example, in the MU task, participants memorize a series of numbers in parallel while applying a sequence of arithmetic operations to them. However, such explicit approaches for assessing WM capacity are time-consuming, stressful, and not well-integrated into HCI applications.
WM capacity of a user will be reflected on her or his behaviour while interacting with a computer. For modeling user behaviour in HCI, Recurrent Neural Networks, especially Long Short term Memory (LSTM) networks, have been widely adopted. Wöllmer [5] introduced an LSTM-based verbal and non-verbal behavior analysis for inferring the spoken content as well as the user's effective state from speech and video signals. Baccouche et al. [6] proposed a sequential model

*This work has been done within the project DINCO "Detection of Interaction Competencies and Obstacles". We thank the German Research Foundation (DFG) for funding DINCO project under the reference number PU 613/1-1.

for human action recognition. Their model works in two steps: first, it learns spatio-temporal features using the extension of Convolutional Neural Networks to 3D. Second, a RNN is then trained to classify each sequence considering the temporal evolution of the learned features for each time-step. Alahi et al. [7] analyzed pedestrians' trajectories, predicted by LSTM, to demonstrate the motion behaviour learned by their LSTM model. Putze et al. [8] introduced an LSTM model to detect memory-based interaction obstacles from behavioural data observed during HCI contexts. However, the obstacle which had been considered is a memory-loading secondary task that deteriorates the performance of the HCI main task regardless of individual WM capacity.
In this paper, we present an LSTM model for classifying WM capacity from behavioral data during HCI. The first contribution is that we exploit behavioral patterns from input log files of natural HCI situations without using any additional sensors such as cameras, EEG electrodes etc. The second contribution is that to deal with limited amounts of available behavioral data, we present an approach to use a cognitive user simulation to generate additional training episodes.

2 Data Collection

A user study was designed to gather two types of data from participants: MU data as reference labels and behavioral data from an HCI task for classification. The chosen HCI task is a game of Matching Pairs, in which the player reveals hidden cards, two per round, to find pairs of corresponding cards which are then removed. Matching Pairs is the model of a task which relies strongly, but not exclusively, on WM. As differences in WM capacity are more likely to be revealed when WM demand is high, we added a secondary task of serial addition. During playing the game, the sequence of selected cards has been recorded.
In total, 24 subjects (16 male, age from 19 to 48) participated in our experiments. The data collection was approved by the ethics committee of the University of Bremen. Participants achieved a median MU score of 88% (standard deviation of 0.17%). This is substantially above the average MU score of 64%[1] [4] for the general population, showing that discrimination of WM capacity will be challenging due to compressed bandwidth of WM capacities. Participants were then partitioned into two groups, namely MU+ for participants having scores greater than median (11 participants) and MU- for participants having scores less then median (10 participants). Three participants with exactly the median score were excluded from further analysis.

3 Classification Setup

For estimating WM capacity, we implemented a classifier to predict group assignment (MU+ or MU-) based on the recorded behavioral data, represented as

[1] 64% is the weighted average of MU scores of three experiments with 110, 114 and 160 participants respectively.

prefixes of playing sessions of fixed length. Shorter prefixes are preferred as they allow a faster WM capacity assessment. For this purpose, we compare two classification models: A sequential neural model based on LSTMs and a static baseline model based on Linear Discriminant Analysis (LDA). A sequential model of user behavior is a plausible choice as it is able to capture the context of individual user behaviours, e.g. whether the user exploits opportunities to reveal pairs, even if the corresponding cards were revealed long ago. As the number of available training episodes is very limited (each session yields one episode, either for class MU+ or class MU-), we need to generate additional training data. While this is a standard Machine Learning approach for static data (often called *oversampling*) using algorithms such as SMOTE [9] or ADASYN [10], it is more challenging for sequential data: generated samples have to preserve the typical temporal relationships of the original data, i.e. we cannot simply generate each time slice independently. Therefore, we use a cognitive user simulation to generate additional training data. The simulation maintains plausible temporal relationships between time slices in reference data sequences. In the following subsections, we describe the user simulation and the classifier setup.

3.1 Cognitive User Simulation

The Cognitive Memory Model [11] (CMM) is a general computational cognitive model of human memory inspired by the ACT-R theory [12]. The CMM models the activation and retrieval probabilities of stored items (revealed cards in our case). The CMM can be used to implement a generative model of playing Matching Pairs by weighting the need for exploration of unknown cards and for exploitation of known pairs [13]. The CMM has a number of free parameters, such as the degree of memory decay, which determine its predictions of memory performance. Those parameters can be optimized by a genetic optimization algorithm in CMM to best fit the empirical training data. The genetic optimizer maintains a population of CMM configurations consisting of all free parameters. The population is initialized randomly and then repeatedly updated according to the standard operations of genetic optimization, mutation and selection [14]. For selection, we employ similarity metrics to compare a set of generated game sessions to real game sessions, e.g. the matching-pairs statistic as a similarity metric counts the number of removed card pairs per turn. A configuration is then selected more likely if it generates game sessions which are close to the real data according to the specified similarity metrics. To achieve enough variance in the training data, we do not pool all available real sessions together to optimize a global parameter set; instead, we repeat the process for each session individually and only pool the resulting simulated sessions together to form a richer, more varied corpus of simulated sessions.

3.2 Classifiers

For modeling sequential behavioural data, we employ an LSTM network. LSTMs consist of so-called LSTM cells which explicitly store, retain, or forget informa-

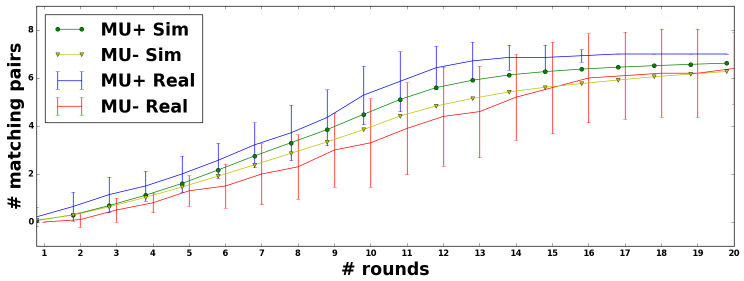

Fig. 1: Mean matching-pairs statistic for MU+ and MU- clusters of real data and simulated data (*Sim*). For real data, whiskers show standard deviation.

tion from previous time steps. This approach is chosen to battle the challenge of vanishing gradient in traditional RNNs. The network for our approach starts with an LSTM layer with 32 LSTM cells as input layer to handle the training data sequences. Sequence items are passed consecutively to LSTM cells as consecutive time steps. The LSTM cells are interconnected where the output of one cell acts as an additional input to the next cell. Such a specific structure allows LSTM to store and retain information from previous time steps. The LSTM layer is followed by a Dropout layer with a rate of 0.5 acting as a regularization technique that avoids over-fitting by randomly ignoring neurons during training [15]. Finally, the output layer is a fully connected dense layer that uses soft-max activation and outputs a binary label. To fit the model, we perform 500 epochs of Stochastic Gradient Descent with adaptive learning rate, $lr = \frac{0.1}{\#epoch}$. The input consists of the ordered sequence of consecutively chosen cards. Each card is represented by two features: The first one encodes its position in the revealing order of motives; the second feature encodes the position of the card in the corresponding pair. Example: the first revealed card is always encoded with the feature vector (1,1), If the second revealed card shows the same motive, it is encoded with the vector (1,2) (second card of the first motive), otherwise with the vector (2,1) (first card of the second motive). The advantage of this feature representation is that it is invariant to the actual motives but still captures the temporal relationships within a sequence. In addition to the raw behavioural data, we manually define statistical features, which are expected to differ between classes, for enriching the input training data. The following features summarize, for a game prefix, how efficient and close-to-optimal a player performed: number of cards left in the game, number of never revealed cards in the game, maximum number of times revealing the same card and number of rounds since game completion (= 0 if game not yet completed). For the LSTM, we calculate the features incrementally for each time step. After 500 training epochs on simulated data, we retrain the resulting model with the original 14 real training sessions. The parameters of the CMM are optimized on those real training sessions as well.

In contrast to the LSTM model, the LDA model considers a whole game at once. It only uses the manually defined features as one vector calculated after the end of the game (prefix). From these features, we train an LDA model (as baseline model) to classify logged games into MU+ and MU- labels.

4 Evaluation

For evaluation, we randomly split the game sessions into 14 sessions for training and 6 for testing. Training and testing data are balanced for classes MU+ and MU-. Splitting is done randomly and repeated for 20 splits, the results are averaged across splits. For each split, we again performed 20 inner iterations with different randomly generated sets of 14000 training episodes (1000 from each real training sesion). We test the trained classification model on the testing data of the respective split and perform analysis for game prefixes of different fixed lengths. We look at game prefixes of length 5, 7, 10 and 15[2]. As performance metric, we report classification accuracy, summarized in Figure 2. These results

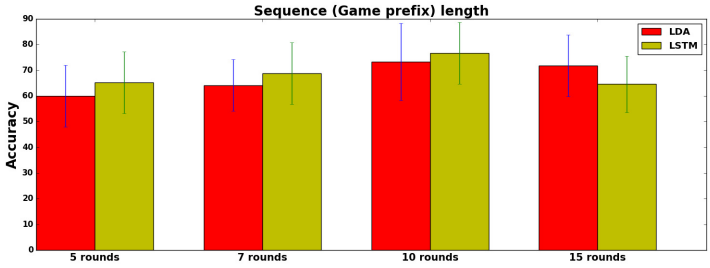

Fig. 2: Average accuracy (and standard deviation) for LSTM and and LDA classification models on real data for game prefixes of different lengths.

show that for game prefix lengths 5, 7 and 10 both LDA and LSTM exploit the richer information of longer game prefixes as accuracy increases with the length of game prefix. In addition, the results show that LSTM outperforms LDA for short sequences. The improvement is statistically significant especially for short game prefixes 5 and 7 ($p = 0.001$, $p = 0.006$ and $p = 0.1$ for 5, 7 and 10 respectively, calculated using a paired t-test on the results of individual iterations). This can be interpreted as LSTM exploits, besides the incremental manual features, temporal dependencies in input sequential data, whereas LDA uses only the static manual features. However, most of the participants finished the Matching Pairs game before 15 rounds, i.e. raw sequential data contains no additional information for the last rounds of such games. This explains why LSTM accuracy decreases for game prefix length of 15. In addition, Figure 1 shows that the difference between MU+ and MU- is largest between 10 and 15 rounds, which makes the discrimination easy for the LDA model. Figure 1

[2]A game prefix with 5 turns lasts 53 seconds on average.

also shows a difference between MU+ and MU- behaviours for shorter prefixes, although standard deviation is high. This implies that partitioning our data into MU+ and MU- is feasible, even for short sequences, but predicting WM capacity cannot be immediately done from simple manual features. For these (most important) cases, the ability of the LSTM model to learn more complex temporal relationships results in improved performance.

References

[1] Andrew RA Conway, Michael J Kane, and Randall W Engle. Working memory capacity and its relation to general intelligence. *Trends in cognitive sciences*, 7(12):547–552, 2003.

[2] John Sweller, Paul Ayres, and Slava Kalyuga. *Cognitive load theory*, volume 1. Springer, 2011.

[3] C. Herff, O. Fortmann, Chun-Yu Tse, Xiaoqin Cheng, F. Putze, D. Heger, and T. Schultz. Hybrid fNIRS-EEG based discrimination of 5 levels of memory load. In *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 5–8, 2015.

[4] Stephan Lewandowsky, Klaus Oberauer, Lee-Xieng Yang, and Ullrich KH Ecker. A working memory test battery for matlab. *Behavior Research Methods*, 42(2):571–585, 2010.

[5] Martin Wöllmer. *Context-sensitive machine learning for intelligent human behavior analysis*. PhD thesis, Universitätsbibliothek der TU München, 2013.

[6] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011.

[7] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016.

[8] Felix Putze, Mazen Salous, and Tanja Schultz. Detecting memory-based interaction obstacles with a recurrent neural model of user behavior. In *Proceedings of the Companion Publication of the 2018 International Conference on Intelligent User Interfaces Companion*, IUI '18 Companion, Tokyo, Japan, 2018. ACM.

[9] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[10] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *IEEE International Joint Conference on Neural Networks*, pages 1322–1328, 2008.

[11] Robert Pröpper, Felix Putze, and Tanja Schultz. Jam: Java-based associative memory. In *Proceedings of the Paralinguistic Information and its Integration in Spoken Dialogue Systems Workshop*, pages 143–155. Springer, 2011.

[12] John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An Integrated Theory of the Mind. *Psychological Review*, 111(4):1036–1060, 2004.

[13] Felix Putze, Tanja Schultz, Sonja Ehret, Heike Miller-Teynor, and Andreas Kruse. Model-based evaluation of playing strategies in a memo game for elderly users. In *Systems, Man, and Cybernetics (SMC), IEEE International Conference on*, pages 929–934. IEEE, 2015.

[14] David B Fogel. An introduction to simulated evolutionary optimization. *IEEE transactions on neural networks*, 5(1):3–14, 1994.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.