Removal of Information from Working Memory

Ullrich K. H. Ecker (ullrich.ecker@uwa.edu.au)
School of Psychology, University of Western Australia

Stephan Lewandowsky (stephan.lewandowsky@bristol.ac.uk)
Department of Experimental Psychology, University of Bristol
School of Psychology, University of Western Australia

Klaus Oberauer (k.oberauer@psychologie.uzh.ch)
Department of Psychology, University of Zurich

Abstract

Standard working memory (WM) updating tasks confound updating requirements with generic WM functions. We introduce a method for isolating a process unique to WM updating, namely the removal of no-longer relevant information. In a modified version of an established updating paradigm, to-be-updated items were cued before the new memoranda were presented. Longer cue-stimulus intervals—that is, longer removal time—led to faster updating, showing that people can actively remove information from WM. Well-established effects of item repetition and similarity on updating RTs were diminished with longer removal time, arguably because representational overlap between out-dated and new information becomes less influential when out-dated information can be removed prior to new encoding. The benefit of removal time was found only for partial updating, not for complete updating of entire memory sets. We conclude that removal of out-dated information can be experimentally isolated, and that removal is a unique, active WM updating process.

Keywords: Working memory updating; Executive functions

Imagine you ask a colleague for his phone extension and he replies: "It's 3266. No, hang on, in my new office it's 3257." Ideally, one should easily discard the last two digits of the outdated information (i.e., "66") and replace them in working memory (WM) with the correct digits (i.e., "57"). However, this updating of WM content is no trivial task, and outdated information often continues to affect WM (Oberauer, 2001).

WM updating has been identified as one of three primary executive processes (Miyake, Friedman, Emerson, Witzki, & Howerter, 2000). Updating has been claimed to be the only executive process to predict fluid intelligence (Friedman et al., 2006). However, most updating tasks used in previous research (e.g., Miyake et al., 2000) not only require WM updating but arguably also measure general WM abilities. This has led some to conclude that updating tasks constitute reliable assays of general WM capacity (Schmiedek, Hildebrandt, Lövdén, Wilhelm, & Lindenberger, 2009).

This creates an unsatisfactory situation. If WM updating tasks measure just the same as other WM tasks such as complex span tasks, then why call them updating tasks? Both conceptually and theoretically, updating can be distinguished from maintenance and processing in WM. If updating is to be established as a non-redundant construct, it must be isolated and measured separately from other WM processes.

In a recent individual-differences study, we identified a processing component that was independent of general WM capacity and unique to situations that demanded WM updating (Ecker, Lewandowsky, Oberauer, & Chee, 2010). In that study we analyzed the processing components involved in widely used WM updating tasks, and we identified three separable components: retrieval, transformation, and substitution. Ecker et al. found that retrieval and transformation operations co-varied with general WM capacity, but that the substitution component did not. We thus argued that substitution is the only process that uniquely represents WM updating, without being "contaminated" by any association with WM. One implication of this analysis is that previous studies measuring WM updating did not separate variance unique to updating from the variance of generic WM processes.

In this article, we further decompose the components of WM updating. In Ecker et al. (2010), we suggested that information substitution can be further subdivided into the *removal* of outdated information and the *encoding* of new information. As encoding is a simple and generic operation involved in many cognitive tasks, we argue that it is the removal process that lies at the heart of memory updating. Accordingly, we focus on the removal of information from WM. Here we show that removal of outdated information can be separated experimentally from encoding of new information.

Our removal measure is based on the work of Kessler and Meiran (2008). In the updating paradigm they used, items (e.g., letters or digits) are presented in a set of individual frames. Items are then repeatedly updated by presenting new items in some frames. On each updating step, between one and n items are updated, where n is the memory set size (equal to the number of frames). Participants had to press a key at the end of each step to indicate that they finished updating.

Kessler and Meiran (2008) proposed a distinction between local and global updating. Local updating refers to changes made to individual items, whereas global updating refers to the integration of all items in the current memory set after individual items were changed. A key piece of evidence for this distinction comes from the observation (Experiment 3 in Kessler & Meiran, 2008) that updating RTs increased with the number of to-be-updated items up to $n-1$ items, but updating was much faster again when all n items were to be replaced on a given step. Thus, updating latencies depended in a non-monotonic fashion on the number of to-be-updated items. Kessler and Meiran (2008) explained this non-monotonicity by assuming that partial updates require a complex sequence of (1) unbinding of the integrated representation of the previous memory set, (2) substitution of some but not all items (i.e., the actual local updating), followed by (3) re-binding the new set as part of the global updating process. In contrast, when the entire set is updated, steps (1) and (2) can be omitted, the old set is simply discarded and a new memory set is encoded and globally updated.

Our interpretation of the non-monotonicity of updating latencies is a specific instantiation of the ideas of Kessler and Meiran (2008), motivated by a computational model of WM, SOB (Lewandowsky & Farrell, 2008). SOB is a two-layer neural network in which items (represented in one layer) are associated to position markers (represented in the other layer) through Hebbian learning, which rapidly modifies the matrix of connection weights between the two layers. In the present updating paradigm, the position marker would represent the location of the item's frame on the screen. Forgetting in SOB is entirely based on interference; there is no time-based decay. To avoid overloading of the system in the absence of decay, an interference model requires a mechanism to remove outdated information; such a mechanism is implemented in the most recent version of SOB (see Oberauer, Lewandowsky, Farrell, Jarrold, & Greaves, 2012). Removal of a specific item involves retrieving that item by cueing with its position marker, and "unlearning" the association between that item and its position. Unlearning is computationally implemented as Hebbian anti-learning. Thus, in SOB the idea of "unbinding" (cf. Kessler & Meiran, 2008) refers specifically to the unbinding of selected items from their position markers. Both encoding and removal of individual items take time (cf. Oberauer, 2001). By contrast, wholesale removal of an entire memory set can be achieved by simply resetting the entire weight matrix, which we assume to be a very rapid process. This explains why updating the entire memory set is faster than partial updating.

In SOB, removal of old information and encoding of new information are described as two separate processing steps. It follows that updating should be facilitated if a cue about what information needs to be removed is given ahead of the to-be-encoded new information. In standard WM updating tasks—including the one used by Kessler and Meiran (2008)—removal can only begin when the new item(s) are presented. For example, when updating the telephone extension from from 3266 to 3257, one can only begin removing the "66" when given the "57". Hence updating times in such a task will include both time for removal and time for encoding. In our new updating task we present cues indicating which items are to be updated before presenting the new to-be-encoded stimuli. People can use the cue only to selectively remove old items from the memory set, not to encode new information. By varying the cue-stimulus interval, we vary the available time for removal. If longer available removal time leads to faster updating RTs, people must have used the cue-stimulus interval for removal or unbinding. We tested this idea in a series of four experiments.

Experiment 1

In Ecker et al. (2010), we found that repeating (i.e., maintaining) an item during an updating task carries a benefit of nearly 400 ms. Experiment 1 tested the idea that this benefit should diminish when people are given the opportunity to remove outdated information before encoding the (identical) updated item.

Method

Experiment 1 used a letter updating task in which each trial consisted of an encoding stage, an updating stage with multiple updating steps, and a final recall stage. Participants encoded 3 letters, presented simultaneously in individual frames. This was succeeded by an unpredictable number of updating steps; each updating step involved only a single, randomly selected letter. In most cases, the outdated letter was replaced with a new letter, but sometimes the letter repeated. Each update was cued before the new letter was presented (the respective frame turned red and bold). Presentation time of this cue—henceforth referred to as removal time—was either 200 ms or 1500 ms. The longer cue should be sufficient time for removal, whereas the shorter cue should be just enough time to focus attention on the to-be-updated frames without permitting removal. After each updating step, the frames were blanked for 500 ms or 1800 ms in the long and short removal-cue condition, respectively, ensuring equal retention intervals in both conditions. The experiment had a 2 (repetition no/yes) $\times$ 2 (removal time: short/long) within-subjects design.

Participants We tested 15 participants, mainly students from the University of Western Australia (UWA).

Apparatus & Procedure The experiment was controlled by a MatLab program. There were 32 trials, each featuring on average 9 updating steps (the number of updating steps ranged from 1 to 21). The probability of item repetition during updating was set at $p = 0.15$. We chose this low rate of repetition to ensure that removal of outdated information was still an attractive strategy. At each updating step, participants were required to press a key when they had finished updating (max. response time was 5 s). This updating RT was the dependent variable of main interest. At the end of each trial, participants were prompted to recall all three letters in random order.

Results

Updating response time data are shown in Figure 1.

A 2×2 repeated measures ANOVA on updating RTs with the factors repetition (no/yes) and removal time (short/long) yielded a main effect of repetition, $F(1,14) = 15.23$, $MSE = 0.03$, $p < .01$, $\eta_p^2 = 0.52$, a main effect of removal time, $F(1,14) = 9.78$, $MSE = 0.01$, $p < .01$, $\eta_p^2 = 0.41$, and crucially, a significant interaction, $F(1,14) = 8.22$, $MSE = 0.01$, $p = .01$, $\eta_p^2 = 0.37$. The interaction indicates that the effect of repetition on updating RTs is larger with short removal (265 ms) than long removal time (86 ms).

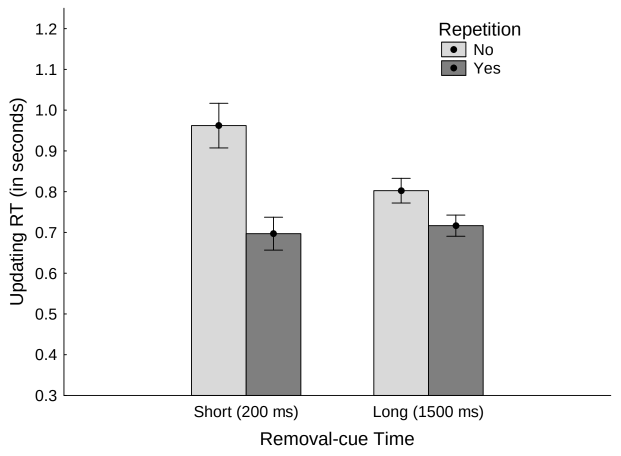

Figure 1: Updating response times from Experiment 1. Vertical bars denote within-subject standard errors of the mean.

Discussion

Experiment 1 demonstrated that the benefit of item repetition during memory updating (Ecker et al., 2010) is strongly reduced if participants are given sufficient time to remove outdated information prior to the encoding of the updated information. In a sense, if there is no time for removal, a condition with repeating items does not require true memory updating—the established item representation in WM can be maintained, no information needs to be removed and substituted—hence the time advantage of repetition. In contrast, to the degree that an item is removed from WM, the time taken to encode a new item into that position will no longer depend on the identity of the removed item. This result pattern supports our notion that active removal of information is integral to WM updating under normal conditions (i.e., when there is no opportunity to "outsource" the removal process.

Experiment 2

Experiment 2 had a similar rationale. Previous research (Lendinez, Pelegrina, & Lechuga, 2011) had shown that updating numbers is quicker when the new to-be-remembered number is similar to the outdated number. Experiment 2 tested whether this benefit would diminish if participants were given sufficient time to remove the outdated number before encoding the new number.

Method

The task in Experiment 2 was very similar to Experiment 1, but it used two-digit numbers. During updating, about half the updates used similar and dissimilar numbers, respectively. This was achieved by manipulating both the proximity of numbers (proximal/distant) and the repetition of one of the digits (yes/no). This resulted in four updating conditions, which had different probabilities of occurrence: proximal/repeating (e.g., updating from 18 to 19; $p = .15$), proximal/non-repeating (e.g., updating from 20 to 18; $p = .15$), distant/repeating (e.g., updating from 59 to 19; $p = .15$), and distant/non-repeating (e.g., updating from 18 to 59; $p = .50$). Proximal updates ranged from -3 to $+3$ (excluding zero); distant/repeating updates were constrained to multiples of 10, and distant/non-repeating updates used prime numbers from 13 to 83. Filler updates with intermediate proximity ($\pm 4 - 8$) were used with $p = .05$. Positive and negative updates were randomly intermixed (this was not considered an experimental factor). Removal time was again an additional factor (short/long).

Participants We tested 27 UWA students.

Apparatus & Procedure The apparatus and procedure was identical to Experiment 1, with the exception that the experiments had 36 trials.

Results

Updating response time data are shown in Figure 2.

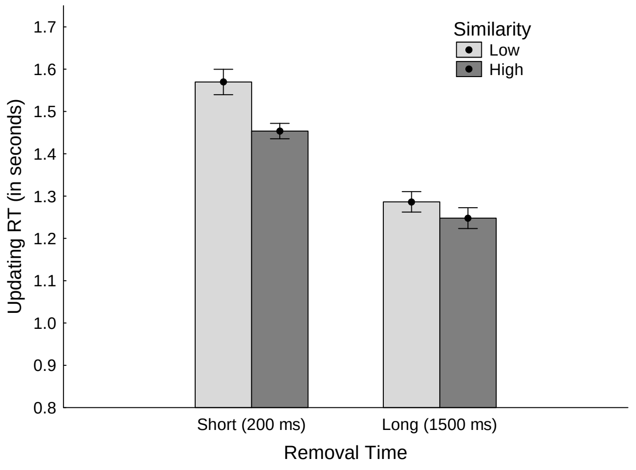

Figure 2: Updating response times from Experiment 2. Vertical bars denote within-subject standard errors of the mean.

For the sake of simplicity, we collapsed the three 'similar' conditions into one, and ran a 2×2 repeated measures ANOVA with the factors similarity (low/high) and removal time (short/long). We found reliable main effects of similarity, $F(1,26) = 11.38$, $MSE = .01$, $p < .01$, $\eta_p^2 = 0.30$, and removal time, $F(1,26) = 101.43$, $MSE = .02$, $p < .001$, $\eta_p^2 = 0.80$. Most importantly, these main effects were qualified by a significant interaction: the similarity effect was larger with short removal time (116 ms) than long removal time (38 ms), $F(1,26) = 6.04$, $MSE = .01$, $p = .02$, $\eta_p^2 = 0.19$.

Discussion

Experiment 2 demonstrated that a well-documented similarity effect in memory updating is diminished when participants are given time to remove a to-be-updated item from memory before encoding the new item. This supports our notion of removal: We assume that the similarity effect in memory updating arises because of representational overlap between the replaced and the new item. That is, two similar numbers share a digit and/or a region in number space, and to the degree that only new features are substituted, not entire item representations, this similarity will facilitate updating. Yet, the more an item representation is removed before the updated number can be encoded, the less facilitation there will be.

Having established some support for our notion of removal, we now turn to a test of our prediction that an active removal process is only utilized during partial updates, not global updates (which unlike partial updates can be achieved more efficiently by 'wiping' memory, or in SOB terms, by resetting the weight matrix).

Experiment 3

Based on the distinction between slow selective removal and fast resetting of the weight matrix, we predicted that longer removal times should lead to a substantial time gain in partial updating, but little gain on updating steps replacing the entire memory set.

In Experiment 3, the letter updating task was modified such that each updating step could update the memory set either partially or entirely (i.e., 1-, 2-, or 3-frame update).

Methods

Participants Sixty-nine UWA undergraduates participated.

Apparatus and Procedure Apparatus and procedure were identical to previous experiments except that on each step, new letter(s) were presented in 1, 2, or 3 frames, and that there were 28 trials in total.

Results

Updating response time data are shown in Figure 3.

A two-way repeated measures ANOVA on updating response times yielded a significant main effect of the number of updated frames, $F(2,136) = 64.30$, $MSE = 0.03$, $p < .001$, $\eta_p^2 = 0.49$, a significant main effect of removal time, $F(1,68) = 281.68$, $MSE = 0.02$, $p < .001$, $\eta_p^2 = 0.81$, as well as a significant interaction, $F(2,136) = 109.55$, $MSE = 0.01$, $p < .001$, $\eta_p^2 = 0.62$. Planned contrasts showed that on average it took significantly longer to update two frames compared to one (1.44 seconds vs. 1.26 seconds; $F(1,68) = 146.29$, $MSE = 0.02$, $p < .001$) and that with 1 or 2 frames, updating took significantly longer with short as compared to long removal time (1.53 vs. 1.17 seconds; $F(1,68) = 320.14$, $MSE = 0.03$, $p < .001$. However, updating three frames was relatively quick (1.23 seconds), and removal time had a negligible impact on updating time when all three frames were updated. While the removal time effect was statistically significant (arguably due to the large sample size and power; 1.26 vs. 1.20 seconds; $F(1,68) = 13.59$, $MSE = 0.01$, $p < .001$), it was much smaller with three frames than with one or two frames (53 vs. 362 ms).

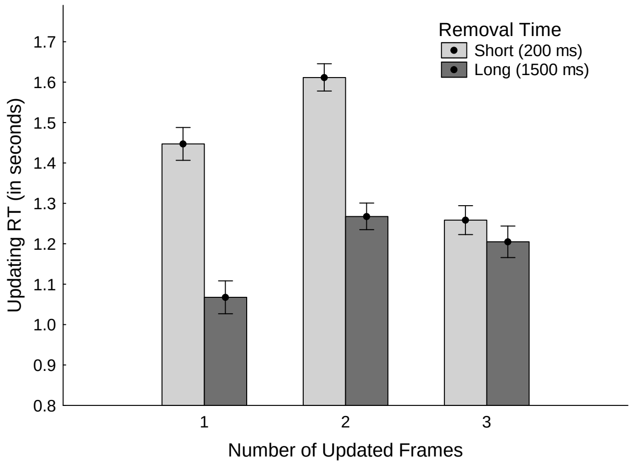

Figure 3: Updating response times from Experiment 3. Vertical bars denote within-subject standard errors of the mean.

Discussion

The results of Experiment 3 support our hypothesis that partial updating of a memory set involves a process of active removal. "Bringing forward" this removal process by cueing the to-be-updated frames prior to presentation of the new items sped up partial updating substantially. This time saving of roughly 300-400 ms can be interpreted as the time that is required to remove information from WM.

In contrast, the opportunity to remove items from memory before presentation of new memoranda brought no substantial advantage when the entire memory set was updated. This finding supports our notion that memory can be cleared almost instantly, and that updating of an entire memory set does not require the time-consuming selective removal process.

The speed-up induced by the long removal cue did not increase with the number of to-be-updated items. This unexpected observation seems to imply that removing one item from WM takes as long as removing two items. There are two possible explanations. One is that removal of multiple items can occur in parallel. While this is a theoretical possibility, it is at odds with SOB's notion that items must be retrieved individually to be removed by anti-learning.[1]

The second explanation is that people use the removal time only to remove one item, even when two items are about to be updated. This might be an efficient strategy because item-specific removal is likely to require the focus of attention, and switching the focus of attention to a new item takes time (cf. Garavan, 1998). Thus, the most efficient use of removal time

[1]This explanation may still warrant further investigation, as unlike our position presented here, people might not use the cue time for a slow removal process, but only to *find* the to-be-removed items. However, this explanation cannot easily account for the reduction of the repetition/ similarity effects observed in Experiments 1 and 2.

might be to focus on the first to-be-removed item and remove it, then wait. As soon as the new item(s) are presented, one of them can immediately be encoded in the currently focused frame. Only then would the focus move on to the second to-be-updated item (if there is one).

This interpretation is in line with a recent proposal by Kessler and Oberauer (2013), namely that partial updating also involves task-switching, which participants may likewise try to avoid. This notion assumes that, without substantial pre-cued removal time, participants scan the items from the beginning of the list to the end, starting in a maintenance mode (M). Hence, if the first item on the list is not updated, people might refresh that representation, then move to the next item. As soon as they encounter an item that requires updating, they switch to updating mode and actively remove that item from WM (U). Updating frames 1 and 3 would hence require 3 switches, (M)UMU; updating frames 1 and 2 two switches, (M)UUM; updating frames 2 and 3 one switch, (M)MUU. In the case of 1-frame updates, updating frame 1, (M)UMM, and updating frame 2, (M)MUM, both require two switches, but updating frame 3 only requires one switch, (M)MMU. A strategy of scanning the list up to the first to-be-updated item, switching to updating mode, removing the item, and waiting would hence minimize both focus-switch and task-switch costs.

We applied multi-level regression analysis to the data of Experiment 3 to confirm the importance of task switching as specified above. We coded processes required at each updating step with the following parameters: The number of items to encode (E; 1-3), the number of items to remove (R; 0-2), and the number of task switches (SW; 0-3). Additional parameters were introduced in the modeling, including a wipe (W) parameter (coding the discarding of an entire memory set), and a refresh (RF) parameter (coding the number of non-updated, to-be-refreshed items). Importantly, coding differed for short and long removal time conditions. For example, with long removal time a two-frame update of frames 2 and 3 would involve no refreshing or switching: Participants could use the removal time to refresh the first item, switch to updating mode for frames 2 and 3, remove the letter in frame 2, then wait to encode the new letter in frame 2, and remove the old and encode the new letter in frame 3 ($RF = SW = W = 0; R = 1; E = 2$). In contrast, if removal time is short, participants would have to refresh the first item, switch once, remove two items and encode the two replacement items ($W = 0; RF = SW = 1; R = E = 2$).

A model including both removal and task switch parameters achieved the best fit; $UpdatingRT = 704 + 156 * E + 87 * R + 393 * SW - 95E * SW$, with a coefficient of determination $COD = 0.437$ and a Bayes Information Criterion $BIC = 23403$. (BIC of the best-fitting model without removal parameter was 23475; according to Raftery (1996) this provides very strong evidence in favor of the removal model).

Experiment 4

Experiment 3 suggested that people only remove one item even with long removal time. We argued this is in fact efficient behavior as it minimizes focus and task switch costs. Yet, an alternative hypothesis is that people would have removed more information had they had more time. Experiment 4 tested the idea that with sufficient removal time, people might remove more than one item, and thus show shorter updating RTs in conditions that would benefit from the removal of more than one item.

Methods

Participants We tested 34 UWA undergraduates.

Apparatus and Procedure Apparatus and procedure were identical to Experiment 3 except for the addition of a third, very-long removal time condition (i.e., removal time conditions of 200, 1500, and 3000 ms), and the omission of the full update condition (i.e., either 1 or 2 frames were updated with each updating step). There were 15 trials in total.

Results

Updating response time data are shown in Figure 4.

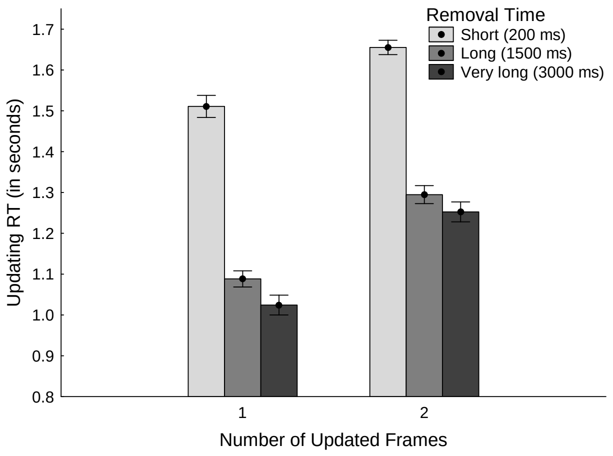

Figure 4: Updating response times from Experiment 4. Vertical bars denote within-subject standard errors of the mean.

A 2×3 repeated measures ANOVA on updating RTs with the factors number of updated frames (1 vs. 2) and removal time (short/long/very long) yielded a main effect of frame number, $F(1,33) = 71.66$, $MSE = 0.03$, $p < .001$, $\eta_p^2 = 0.68$, a main effect of removal time, $F(2,66) = 167.88$, $MSE = 0.02$, $p < .001$, $\eta_p^2 = 0.84$, but no significant interaction, $F(2,66) = 2.06$, $MSE = 0.02$, $p > .10$, $\eta_p^2 = 0.06$. Results showed that it took longer to update two frames than one, and that more removal time led to faster updating RTs. However, the effect of removal time was strong when comparing the short (200 ms) and long (1500 ms) conditions, but negligible when comparing long and very-long (3000 ms) conditions. The lack of interaction means that doubling removal time did not lead to quicker updating, not even in the 2-frame updates where performance could have benefitted from the removal of both to-be-updated items.

Applying the regression models (as specified in Exp. 3), we found that a simple model provided the best fit: $UpdatingRT = 901 + 131 * R + 255 * SW$, with $COD = 0.38, BIC = 6865$. (Note that the more complex model specified in Exp. 3 did explain more variance than this simple model when fit to the data of Exp. 4, but had a higher overall BIC. The BIC for the best removal-free model was 6880; this is strong evidence in favor of the simple removal model.)

Discussion

Experiment 4 showed that in the present task people only removed one item in anticipation of an update, even when this update concerned more than one item. This supports our notion that people avoid focus and task switching when removing information from WM during updating.

General Discussion

In this article we have introduced a novel measure of WM updating. Traditional WM updating tasks arguably measure general WM processes in addition to updating, whereas it is the removal of information from WM that is specific and unique to WM updating. We demonstrated that giving people preparation time to remove information from WM speeds up updating when new information is subsequently presented, but only when a subset of the memory set is updated. Updating an entire memory set does not benefit (much) from preparation time, arguably because—in line with the predictions derived from SOB (Lewandowsky & Farrell, 2008; Oberauer et al., 2012)—the time-consuming removal process only applies to individual items, whereas an entire memory set can be removed by instant resetting of the weight matrix. Our notion of removal by unlearning item-position associations is a specific incarnation of the more general idea advanced by Kessler and Meiran (2008), who suggested that partial updating of memory sets involves "dismantling" or "unbinding" the old representations.

Whereas our research is guided by and supports the SOB model, it is important to note that other researchers have provided independent evidence for the existence of an active and attention-demanding removal process. For example, Fawcett and Taylor (2012) have shown that directed forgetting of an item (1) slows down responses on an unrelated secondary task for up to 2.6 seconds, and (2) impairs incidental memory for a subsequent distractor, in particular when the directed forgetting of the studied item is successful.

We assume removal to be crucial to maintaining a functional WM system that can efficiently focus on relevant information. Further research is needed to ascertain whether removal abilities co-vary with WM capacity, whether they predict intelligence-related variance, or whether they relate to other executive functions.

Acknowledgments

This research was supported by Australian Research Council Discovery Grants and Research Fellowships to UE and SL.

References

Ecker, U. K. H., Lewandowsky, S., Oberauer, K., & Chee, A. E. H. (2010). The components of working memory updating: An experimental decomposition and individual differences. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *36*, 170–189.

Fawcett, J. M., & Taylor, T. L. (2012). The control of working memory resources in intentional forgetting: Evidence from incidental probe word recognition. *Acta Psychologica*, *139*, 84-90.

Friedman, N. P., Miyake, A., Corley, R. P., Young, S. E., DeFries, J. C., & Hewitt, J. K. (2006). Not all executive functions are related to intelligence. *Psychological Science*, *17*, 172-179.

Garavan, H. (1998). Serial attention within working memory. *Memory & Cognition*, *26*, 263-276.

Kessler, Y., & Meiran, N. (2008). Two dissociable updating processes in working memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *34*, 1339–1348.

Kessler, Y., & Oberauer, K. (2013). *Working memory updating latency reflects the cost of switching between maintenance and updating modes of operation.*

Lendinez, C., Pelegrina, S., & Lechuga, T. (2011). The distance effect in numerical memory-updating tasks. *Memory and Cognition*, *39*(4), 675-685.

Lewandowsky, S., & Farrell, S. (2008). Short-term memory: New data and a model. In B. H. Ross (Ed.), *The psychology of learning and motivation* (Vol. 49, pp. 1–48). London, UK: Elsevier.

Miyake, A., Friedman, N. P., Emerson, M. J., Witzki, A. H., & Howerter, T. D., A.and Wager. (2000). The unity and diversity of executive functions and their contributions to complex "frontal lobe" tasks: A latent variable analysis. *Cognitive Psychology*, *41*, 49-100.

Oberauer, K. (2001). Removing irrelevant information from working memory: A cognitive aging study with the modified Sternberg task. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *27*, 948–957.

Oberauer, K., Lewandowsky, S., Farrell, S., Jarrold, C., & Greaves, M. (2012). Modeling working memory: An interference model of complex span. *Psychonomic Bulletin & Review*.

Raftery, A. E. (1996). Approximate bayes factors and accounting for model uncertainty in generalised linear models. *Biometrica*, *83*(2), 251-266.

Schmiedek, F., Hildebrandt, A., Lövdén, M., Wilhelm, O., & Lindenberger, U. (2009). Complex span versus updating tasks of working memory: The gap is not that deep. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *35*, 1089–1096.