Karlsruhe Institute of Technology

Modeling Human Memory Performance Under Influence Of Cognitive Workload

Bachelor Thesis at the Cognitive Systems Lab
Prof. Dr.-Ing. Tanja Schultz
Institute for Anthropomatics
Department of Computer Science
Karlsruhe Institute of Technology

by

Lucas Bechberger

Supervisors:

Prof. Dr.-Ing. Tanja Schultz
Dipl.-Inform. Felix Putze

Date of Registration: April 10, 2012
Date of Delivery: August 10, 2012

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 10. August 2012

Abstract

This thesis examines human memory performance under influence of different intensities of cognitive workload. A psychological experiment was conducted using a recognition test and divided attention at encoding time. Two variations of the switching task were used as secondary tasks. The experiment results contain significant effects of different workload modes as well as significant effects within the workload modes. Observed effects include differences between delayed and immediate retrieval, between single and multiple encoding and between semantically related and semantically unrelated words. Based on the ACT-R declarative module and LTM^C, a model of human memory performance is devised, using WordNet as database. This model tries to predict the observed effects by using the ACT-R base level activation equation (for delayed vs. immediate retrieval and single vs. multiple encoding) and a spreading mechanism (for semantically related vs. semantically unrelated words). As the evaluation results show, the devised WorkloadMemoryModel is capable of predicting the effects observed regarding response time, hit rate and false alarm rate. Moreover, transferring a memory model to different workload modes by modifying the model's free parameters has proved to be a reasonable approach. Further research is advised, e.g. regarding the integration with other models and systems.

Zusammenfassung

In dieser Bachelorarbeit wird die menschliche Gedächtnisleistung unter dem Einfluss verschiedener kognitiver Auslastungsgrade untersucht. Hierzu wurde ein psychologisches Experiment durchgeführt, das einen Erkennungstest und geteilte Aufmerksamkeit während Lernphasen verwendet. Als Nebenaufgabe wurden zwei Varianten des „switching task“ eingesetzt. Die Ergebnisse dieses Experiments beinhalten sowohl signifikante Unterschiede zwischen den einzelnen Auslastungsgraden als auch singnifikante Effekte innerhalb der Auslastungsgrade. Die beobachteten Effekte umfassen Unterschiede zwischen verzögerter und sofortiger Abfrage, zwischen einmaligem und mehrfachem Lernen und zwischen semantisch zusammenhängenden und semantisch unabhängigen Wörtern. Basierend auf dem deklarativen Modul der kognitiven Architektur ACT-R und dem LTM^C-Gedächtnismodell wird ein Modell menschlicher Gedächtnisleistung entwickelt, welches WordNet als Datenbasis verwendet. Dieses Modell versucht die beobachteten Effekte vorherzusagen, indem es die ACT-R Base-Level-Aktivierung (verzögertes vs. sofortiges Abfragen, einfaches vs. mehrfaches Lernen) und einen Spreading-Mechanismus (semantisch zusammenhängende vs. semantisch unabhängige Wörter) verwendet. Wie die Evaluation dieses Modells zeigt, ist es in der Lage die beobachteten Effekte bezüglich Reaktionszeit, Trefferrate und Falschalarmrate vorherzusagen. Außerdem hat sich die Übertragung eines Gedächtnismodells in verschiedene Auslastungsgrade durch Anpassen der Modellparameter als sinnvoller Ansatz erwiesen. Weitergehende Forschungsarbeiten werden empfohlen, beispielsweise bezüglich der Integration mit weiteren Modellen und Systemen.

Contents

1. Introduction

In recent years, significant progress has been made in Human Computer Interaction (HCI) with user interfaces becoming much easier to use. Researchers have become interested in modeling the user's internal state in order to enable their systems to react appropriately to the user's mood, level of stress, etc. One crucial influence on the user's performance is the current workload: As studies of cognitive psychologists (e.g. Hart & Staveland, 1988 [1] and Bourne & Yaroush, 2003, [2]) have shown, users perform more slowly and less accurately under high workload (e.g. if simultaneously performing a secondary task). This thesis addresses the influence of cognitive workload on human memory performance. It extends the work of different psychologists (e.g. Craik et al., 1996, [3]) by explicitly devising a cognitive model of human memory performance (based on the declarative module of ACT-R) for predicting response time and response quality (measured as hit rate and false alarm rate) under different workload settings. This workload-sensitive memory model serves as basic research. However, with continued research, the results from this thesis will provide an opportunity for user interfaces to predict whether the user still remembers previously presented information or whether this piece of information should be presented again.

1.1 Purpose

The purpose of this thesis is to devise a workload-aware model of human memory that can predict human memory performance in different workload settings. This goal is basically achieved by combining existing models (the ACT-R declarative module and LTM^C) and using WordNet (a linguistic model of the English language) as database. Using a large database like WordNet is more plausible than using only directly task-relevant pieces of information: In the first case, human memory (which can be considered to contain thousands of pieces of information) is modeled more accurately and potential interference effects can be modeled better than in the latter case (see Schultheis et al., 2006, [4] for a similar reasoning).

Prior to devising this model, a psychological experiment is conducted. This is done for two reasons: On the one hand, real world data is needed for optimizing and

evaluating the model. On the other hand, in order to develop a model that can predict human memory performance, this performance must be analyzed beforehand: The knowledge of the effects to model is necessary for modeling them.

As the ACT-R cognitive architecture has proved (see Anderson et al., 2004, [5]), it is possible to devise good models of human memory. However, these models generally do not take into account different workload settings. The most important question to be answered by this thesis is the following: Can models of human memory like the ACT-R declarative module be adapted to different workload settings by modifying the model's free parameters in a plausible way, or is it necessary to devise completely or partially new models for human memory under workload? Moreover, if different workload modes can be simulated by modifying the model's parameters, how do these parameter modifications influence the model's behavior? This thesis will address these questions by devising a model based on the ACT-R declarative module and LTM^C, by optimizing the model's parameters for different workload modes and by subsequently evaluating the optimized parameters and the model's predictions.

If the model devised in this thesis is combined with a workload recognition system (e.g. Heger et al., 2010, [6]) and a workload-aware dialog system (e.g. Putze & Schultz, 2009, [7]), a workload-aware user interface seems to be in reachable distance. Such a workload-aware user interface with the capability to predict the user's memory performance can be of use in situations where a user must make severe decisions in a demanding environment. For example, consider an operator in an emergency call center who has to decide under high pressure when to send how many ambulances or fire trucks to which location. In this case, a system aware of the operator's workload level and how it affects the operator's memory performance can help the operator to make good decisions, for example by re-prompting important information the operator might have forgotten due to high workload.

1.2 Structure

In chapter 2, existing approaches of modeling human memory and related work regarding the influence of cognitive workload on human memory performance are presented.
In chapter 3, a psychological experiment for measuring the influence of cognitive workload on human memory performance is described: The methods are explained, and the experiment's results are presented and discussed.
In chapter 4, a model of human memory performance is devised which can handle different workload modes. Furthermore, an overview of its implementation is given.
In chapter 5, the model devised in chapter 4 is evaluated by systematically trying to reproduce the effects observed in the results of the experiment from chapter 3.
In chapter 6, this thesis is concluded by summarizing the key results and by listing approaches for further research.

2. Principles

This chapter summarizes the principles regarding human memory modeling, cognitive workload, memory tests and divided attention. It reviews related work and describes relevant approaches.

2.1 Models of Human Memory

In his book "Unified Theories of Cognition" (1990) [8], Newell lists several criteria which characterize the human mind. If a cognitive architecture tries to mimic human cognition processes, it has to meet this criteria to be considered human-like. One of this criteria is the ability to store information persistently and the ability to retrieve this stored information at a later point in time. To meet this criteria, cognitive architectures must provide some model of human memory that provides ways of storing ("memorizing") and retrieving ("remembering") information.

In general, two types of memory can be distinguished: *short term memory* (STM) and *long term memory* (LTM). The memory span of STM is considered to be rather short (usually less than a minute), whereas the memory span of LTM is considered to be infinite. Moreover, the capacity of STM is considered to be relatively small, whereas the capacity of LTM is considered to be nearly unlimited.

However, this distinction into STM and LTM is not accepted by all researchers: Some declare *working memory* as a third type of memory and as an intermediate stage between STM and LTM. Others (e.g. Cowan, 1988, [9]) propose the model of one unified memory with STM being the part of memory where the attention is currently directed to.

For the scope of this thesis, the possible division of human memory into STM and LTM is not taken into account. Although the ACT-R declarative module and LTM^C explicitly model human LTM, the model devised in chapter 4 is not assigned to one of those memory types, but is treated as a unified memory model.

Another distinction of human memory can be made regarding its content (see Squire, 1992, [10]): It can be distinguished into *declarative memory* and *procedural memory*. Declarative memory contains explicit knowledge of facts (e.g. "A book consists of

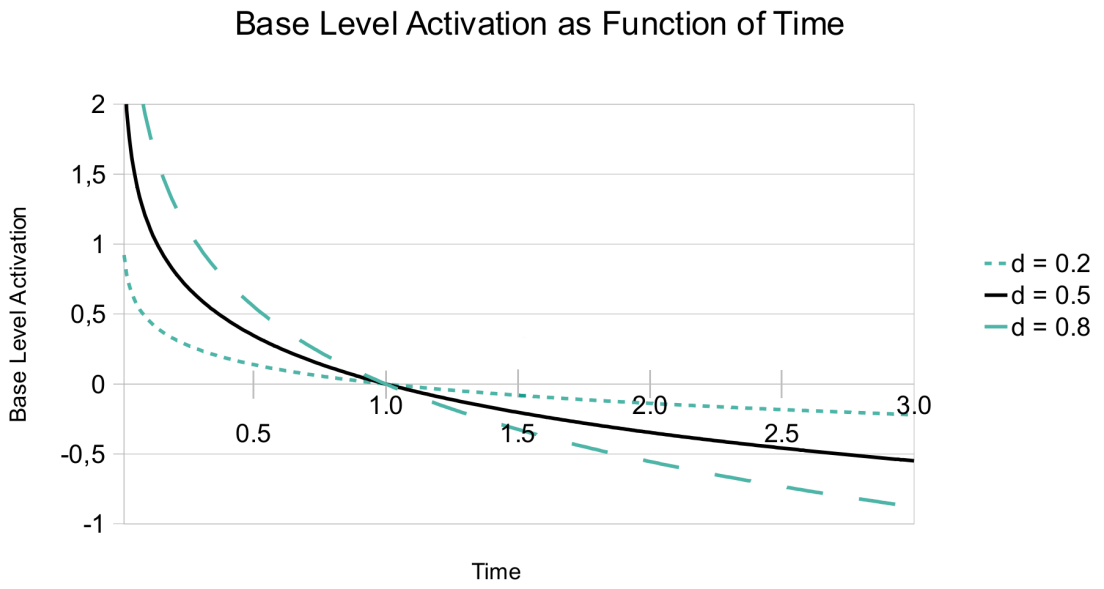

Figure 2.1: This figure shows the ACT-R base level activation as a function of time for three different values of d (0.2, 0.5 and 0.8). It is assumed that the chunk corresponding to this base level activation was reinforced only once at $t = 0$.

pages.") and events (e.g. what happened at your last birthday party), whereas procedural memory contains implicit memory of activities (e.g. how to ride a bicycle). For the scope of this thesis, only declarative memory is considered.

In the following sections, three existing approaches of modeling human memory are described: The ACT-R declarative module, the LTM^C memory model and the WordNet database. Note, that these approaches are closely related: For example, LTM^C was designed based on the ACT-R declarative module, and there have been attempts to integrate WordNet into the ACT-R cognitive architecture (see Emond, 2006, [11]).

2.1.1 The ACT-R declarative module

This section refers to Anderson et al. (2004) [5], where the complete ACT-R (**A**daptive **C**ontrol of **T**hought – **R**ational) cognitive architecture is presented, and to the ACT-R 6.0 Reference Manual [12] and the ACT-R 6.0 Tutorial [13]. In this section, only the ACT-R declarative module, which models human declarative knowledge, will be discussed.

Information in ACT-R is represented in form of so called *chunks*. Each chunk has a type and several attributes. The value of a chunk attribute can be either an atomic value (e.g. an integer) or another chunk. Chunks are used to transfer information between the different modules of the ACT-R cognitive architecture. The ACT-R declarative module maintains a set of chunks that represent the current declarative knowledge. Chunks can be inserted into the declarative module and they also can be retrieved from it.

To model the effects of forgetting, a chunk can only be retrieved with a certain probability. This probability decreases over time, if the given chunk is not reinforced

again. In order to compute the retrieval probability, an *activation* value, which is assumed to track log odds, is assigned to each chunk. This activation not only determines the retrieval probability, but also the response time (which is the time needed to successfully retrieve this chunk from the declarative memory).

For retrieval, the declarative module is provided with a chunk describing the information needed. The declarative module then searches for chunks fulfilling this description and selects the chunk with the highest activation value amongst them for retrieval.

The activation value is composed of three different summands: base level activation, associative activation and noise activation.

Equation 2.1 defines the *base level activation* of a chunk i that was reinforced n times, depending on the current time t_c and the time of the k-th reinforcement t_k. The decay parameter d describes the rate of decay over time.

$$B_i = \ln(\sum_{k=1}^{n} (t_c - t_k)^{-d}) \tag{2.1}$$

The base level activation rises with practice and declines over time (which simulates forgetting). Figure 2.1[1] illustrates the base level activation of a chunk that has been encoded at $t = 0$ as a function of time, using three different values for the decay parameter. As it can be seen in the figure, a higher value of d causes a faster drop of base level activation and is therefore associated with faster forgetting.

Another source of activation is the *associative activation*. The associative activation is computed as in equation 2.2, where W_j reflects the attentional weight of element j at the current point in time and S_{ji} represents the association strength between chunk i and element j.

$$C_i = \sum_j W_j S_{ji} \tag{2.2}$$

W_j is usually set to $1/n$ with n being the number of activation sources. S_{ji} is usually set to $S - ln(fan_j)$ with fan_j being the number of facts that are associated to element j. The parameter S is often set to a value of 2 which has emerged as reasonable value.

The third source of activation is the *noise activation* N_i. It is simply a random variable following a logistic distribution with mean value 0 and variance σ^2.

So the *overall activation* of a chunk in the ACT-R declarative module can be computed as in equation 2.3 by simply summing up the three mentioned sources of activation.

$$A_i = B_i + C_i + N_i = \ln(\sum_{k=1}^{n} (t_c - t_k)^{-d}) + \sum_j W_j S_{ji} + Logistic(0, \sigma^2) \tag{2.3}$$

[1] All figures in this thesis without an explicit reference were created by using OpenOffice.org 3.3.0, Dia 0.97.1 and Gimp 2.6.7.

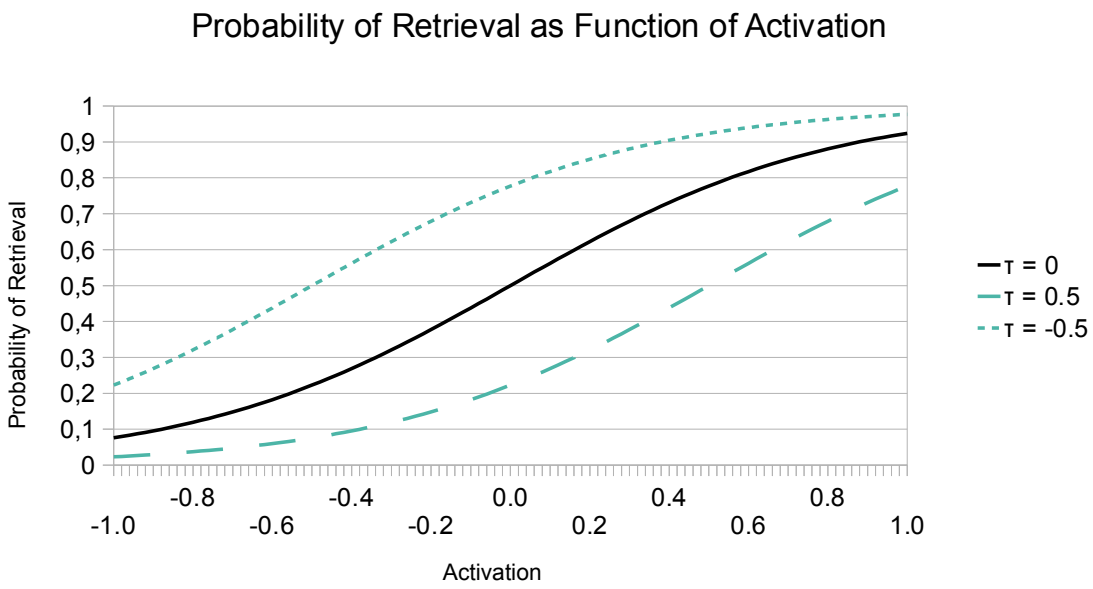

Figure 2.2: This figure shows the probability of retrieval as function of activation. The probability of retrieval is illustrated for three different values of τ. The parameter s is alway set to 0.4 in this example.

As noted before, in the ACT-R declarative module, chunks can only be retrieved with a certain probability. Equation 2.4 shows how the *probability of retrieval* is computed:

$$P_i = \frac{1}{1 + e^{-(A_i - \tau)/s}} \tag{2.4}$$

Chunks can only be retrieved if their activation value is greater than a specific threshold value τ. The parameter s controls the sensitivity of the retrieval probability against varying activation values and is usually set to 0.4. It also represents the amount of noise in the system and has an influence on the variance σ^2 of the noise activation. Figure 2.2 illustrates equation 2.4 for three different values of τ and with $s = 0.4$.

As stated previously, if a chunk is retrieved successfully, the ACT-R declarative module also provides the *latency of retrieval* describing the delay between the request and the retrieval of this chunk. It is computed by using equation 2.5 where A_i is the activation value of chunk i and F is a latency factor.

$$T_i = F \cdot e^{-A_i} \tag{2.5}$$

Although F is another free parameter, a general relationship exists between the latency factor F and the retrieval threshold τ. It can be stated as $F \approx 0.35 \cdot e^{\tau}$. This can be interpreted as the retrieval latency for $A_i = \tau$ being approximately 0.35 seconds.

However, equation 2.5 only holds if a chunk can be retrieved successfully. In case no chunk matches the request or none of the matching chunks has an activation value above threshold level, the *failure latency* is the time until this failure to retrieve is

Figure 2.3: Example for LTM^C: The three nodes linked by two edges represent the knowledge that London is north of Paris (taken from Schultheis et al. (2006) [4])

signaled. According to the ACT-R Tutorial [12][Unit 4] , it can be computed by using equation 2.6:

$$T_i = F \cdot e^{-\tau} \tag{2.6}$$

If encoding and responding are involved (e.g. when reading a word, then trying to remember this word and finally pushing a button to indicate that the word is known), the overall recognition time (measured as delay between beginning of the encoding and the actual response) can be computed by adding an additional parameter I to the retrieval latency as defined in equation 2.5 and to the failure latency as defined in equation 2.6, respectively. This parameter I is the intercept time and reflects the time needed to encode the item and to perform a response. The *recognition time* can then be computed as in equation 2.7.

$$\text{recognition time} = I + T_i \tag{2.7}$$

As described in Anderson et al. (2004) [5] and in Schultheis et al. (2006) [4], the ACT-R declarative module can be used in a broad range of applications and provides good predictions of average human memory performance. However, ACT-R is generally not capable of predicting differences between individuals or caused by different conditions.

2.1.2 LTM^C

Schultheis et al. (2006) [4] evaluated the long term memory (LTM) modules of different cognitive architectures, including the ACT-R declarative module. They find that the ACT-R approach is the most promising one, but they also note that there is still potential for improvement. For example, they criticize the inflexible chunk structure used by the ACT-R declarative module: The chunk type (which is preset by the modeler) determines how retrieved information is grouped. Moreover, only chunks completely fulfilling the chunk description can be retrieved, so partial matching (i.e. retrieving a piece of information that does not completely fulfill the description) is not possible.

To resolve these shortcomings of the ACT-R declarative module, Schultheis et al. propose an improved memory model for cognitive architectures, called LTM^C.

In LTM^C, the atomic memory elements are not chunks, but nodes. Each node has a name and one or more edges linking it to other nodes. These links theirselves do not establish relations – relations are also represented as nodes. See figure 2.3 for an example, how relations between two nodes are encoded in the LTM^C memory model.

It can be easily seen that there are two types of nodes: object nodes (e.g. "London") and relationship nodes (e.g. "north-of"). This allows for example to easily encode that one object is between two others, by having a "between" relationship node that is connected to three object nodes. If relations were represented by edges and only object nodes were used, the representation of this fact would be more difficult since edges can only connect two nodes.

Moreover, an *is-a* relation for establishing hierarchies ("dog – *is-a* – mammal – *is-a* – animal" or "north-of – *is-a* – direction relation") is available. This *is-a* relationship is not represented as a node, but as a special *isa-connection* (which in principle is a special kind of edge) to avoid infinite regress (which would occur when this *is-a* relation was also represented as node).

As in ACT-R, every node has an activation value composed of base level activation, spreading activation and noise activation. Base level activation and noise activation are computed using the same equations as in ACT-R.

To calculate the spreading activation, the following steps are executed:

- First, the spreading activation of all nodes is set to zero.
- Second, nodes representing elements of the current context are stimulated, i.e. their activation is increased. If for example a person is asked to give the direction relation between the two cities London and Paris, the nodes "London", "Paris" and "direction relation" are stimulated. The amount of total stimulation activation is fixed an is equally distributed among the stimulated nodes.
- Third, the inserted stimulation activation is spread to neighboring nodes via the link edges. Let f^i_{act} be the activation node i has received (either by the initial stimulation or by spreading from another node). This activation is added to this node's current activation value. Then, a fraction of f^i_{act} is spread to all neighboring nodes. Each of the neighboring nodes will receive an equal amount of activation of f^i_{act}/m, with m being the number of neighbors of the current node i.

Spreading direction is constrained by the following rule: Activation that has spread upwards in the hierarchy (following a *isa*-connection) is not allowed to spread downwards again and vice versa. Spreading stops, when the amount of activation to spread falls under a specific threshold value. Schultheis et al. set this threshold value to $S_{context} \cdot 10^{-4}$, with $S_{context}$ being the amount of overall context stimulation that has been inserted into the model.

After the spreading activation is computed, base level activation and noise activation are added to each node's spreading activation to get the overall activation value.

As in ACT-R, only nodes with an activation value greater than a certain threshold can be retrieved. This threshold value is defined as the average activation value of all nodes currently in the memory model. Moreover, retrieval results consist of a subset of connected nodes – neither all nodes with activation above the threshold nor only the node with the highest activation are retrieved. A retrieval result must be a set of nodes S_n containing a path $(i, i+1, i+2, \cdots, j-1, j)$ for every pair of

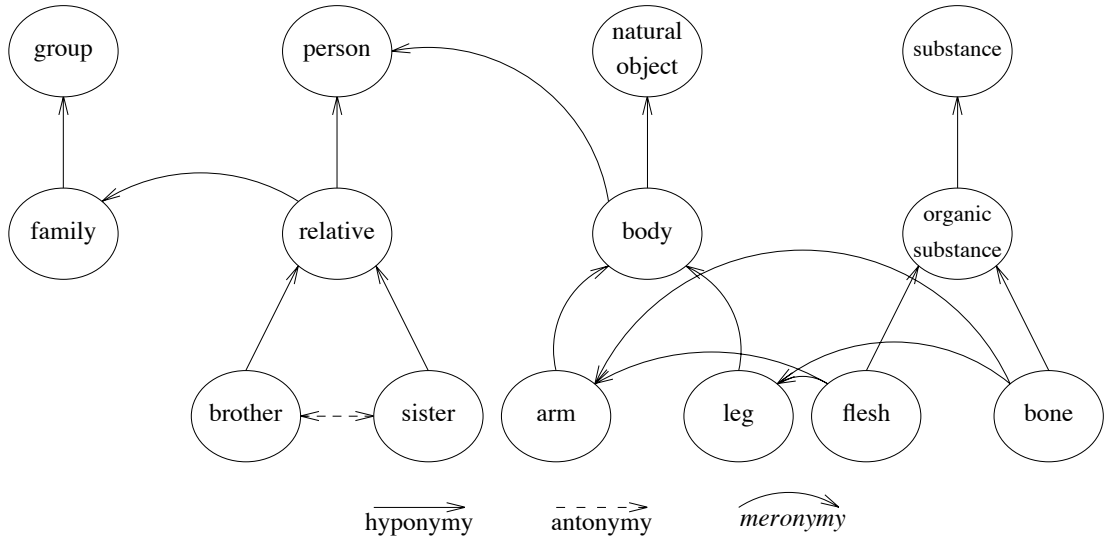

Figure 2.4: This figure illustrates the concept of the different WordNet pointer types. Taken from Miller (1993) [15].

nodes $i, j \in S_n$. The retrieval result is the subset with the highest overall activation amongst the sets of nodes fulfilling this criteria.

2.1.3 WordNet

This section refers to Miller et al. (1993) [14] and Miller (1993) [15].

WordNet is a lexical database for the English language which has been developed at Princeton University. WordNet contains a large number of nouns, verbs, adjectives and adverbs, all grouped into sets of synonyms called *synsets*. For the scope of this thesis, only nouns are considered.

Each synset represents one concept. It consists of a list of words and has a unique identifier. Each synset has also a short descriptive sentence. Words having more than one meaning appear in different synsets. For example the noun "table" belongs to six different synsets. Two of them are {table, tabular array} (description: "a set of data arranged in rows and columns") and {table} (description: "a piece of furniture having a smooth flat top that is usually supported by one or more vertical legs").

Synsets are connected with other synsets via semantic pointers. The relations between noun synsets include hypernymy/hyponymy ("is-a" relation), meronymy/ holonymy ("has-a" relation) and antonymy ("opposite-of" relation). Table 2.1 shows a description and an example for each of the five pointer types. Hypernymy and hyponomy are inverse relations which means that "x is a hypernym of y" implies "y is a hyponym of x" and vice versa. This also holds for meronymy and holomymy. Both hypernymy/hyponymy and meronymy/holonymy are asymmetrical and transitive, whereas antonymy is symmetrical and not transitive. Figure 2.4 illustrates the different pointer types hypernymy, meronymy and antonymy.

By using hypernymy, a hierarchy of noun synsets can be constructed. Note that this hierarchy is not a tree since it is possible for a word to have multiple hypernyms. In

WordNet Pointer Types Between Noun Synsets		
Pointer type	Description	Example
Hypernym	Synset x is a hypernym of synset y if "a y is a kind of x" is true	{tree} is a hypernym of {maple} since "a maple is a kind of tree".
Hyponym	Synset x is a hyponym of synset y if "an x is a kind of y" is true	{maple} is a hyponym of {tree} since "a maple is a kind of tree".
Meronym	Synset x is a meronym of synset y if "an x is a part of y" is true	{stem} is a meronym of {tree} since "a stem is a part of a tree".
Holonym	Synset x is a holonym of synset y if "an x has a y (as a part)" is true	{tree} is a holonym of {stem} since "a tree has a stem as a part"
Antonym	Synset x is an antonym of synset y if "x is the opposite of y" is true	{defeat} is an antonym of {victory} since "defeat is the opposite of victory"

Table 2.1: This table explains and illustrates the five WordNet pointer types regarding noun synsets.

this hierarchy, a hyponym is assumed to inherit all attributes from its hypernym and to add at least one feature that distinguishes this hyponym from its hypernym and all other hyponyms of its hypernym. For example, a {maple} inherits its attributes from its hypernym {tree}, but it can be distinguished from all other types of trees (i.e. all other hyponyms of its hypernym) by the color of its wood etc.

Because of its graph structure (if the synsets are considered as nodes and the relationship pointers between them are considered as edges), WordNet is suited for being the database of a node-based memory model if the memorization of words is of interest.

2.2 Memory Tests

A plethora of tests are available to test a participant's memory performance. Although they differ significantly in many aspects, they share some common principles. This section will describe these common principles and will then present two memory tests in more detail: the California Verbal Learning Test (CVLT) and the Hopkins Verbal Learning Test (HVLT).

2.2.1 Common Principles

In general, memory tests try to measure the performance of declarative memory. This is usually achieved by presenting the participant a specific number of stimuli (e.g. words or images) for memorizing and then testing how many of these stimuli can be recalled.

Usually, a memory test consists of several *encoding phases* and *retrieval phases*. Stimuli are presented for learning during encoding phases and the participant is asked to retrieve them during retrieval phases.

Three main types of retrieval techniques can be distinguished: free recall, cued recall and recognition test.

- In *free recall*, the participant is asked to recall as many stimuli as possible without presentation of any hints. Usually, a time limit is set for free recall. An example would be a participant who has learned the words {*tree, dog, train, air*} and is asked to reproduce as many words as possible.
- In *cued recall*, the stimuli are learned in pairs with one being the cue stimulus and the other one being the target stimulus. During retrieval, the cue is presented and the participant has to recall the associated target. An example would be a participant who has learned the word pairs {*tree - dog, train - air*} and is asked to reproduce the target word for the cue word *tree*.
- In a *recognition test*, the participant is faced with a yes/no question: During retrieval, again stimuli are presented and the participant is asked to decide whether he/she recalls the presented stimulus. An example would be a participant who has learned the words {*tree, dog, train, air*} and is asked whether *train* is one of the words learned before.

Using a recognition test has the advantage that due to its simplicity, the participant's response can be easily recorded (e.g. by logging key presses). On the other hand, because the stimulus is presented again, another reinforcement takes place which possibly biases the results.

Free recall has the advantage that no further reinforcement takes place and so the probability of the results being biased is relatively low. On the other hand, since free recall leaves much freedom to the participant in reproducing the stimuli, the participant's responses are usually harder to record and to compare. Especially exact timings cannot be obtained easily.

Cued recall is somewhere in the middle between free recall and recognition: There is some further reinforcement due to the presentation of cues, but not as intensive as during a recognition test. Moreover, cued recall allows to test for associative memory since participants not only learn stimuli but also associations between those stimuli.

For evaluating a participant's performance, the accuracy (percentage of correct answers) can be computed. In general, recognition has the highest accuracy whereas free recall has the lowest accuracy. This is probably caused by the level of further reinforcement as mentioned above.

2.2.2 The California Verbal Learning Test

The California Verbal Learning Test (CVLT) is a widely accepted test of human declarative memory, regarding memorizing of words. In Delis et al. (1991) [16], the CVLT is described in the following way:

> The CVLT involves the oral presentation of two "shopping" lists (Lists A and B) of 16 words each, with four words each from four semantic categories. Words from the same category are never presented consecutively, which affords an assessment of the degree to which an examinee uses a

> semantic clustering strategy. Immediate free recall of List A is measured for each of five consecutive learning trials. The next trial involves the presentation and immediate free recall of the interference list (List B), followed by "short-delay" free and category-cued recall of List A. Nonverbal testing is then administered for 20 min, followed by the "long- delay" free and category-cued recall trials of List A. A yes/no recognition test of List A is then administered.

As the description shows, the CVLT uses all three types of retrieval techniques: free recall, cued recall and recognition test. In this case, cued recall is not implemented by asking the participant to memorize pairs of words, but rather by giving the category name as a cue. For example, for the words "vest", "sweater", "jacket" and "slacks", the cue would be "cloths" as they all belong to this category. Note that learning and retrieving List B is used as a distractor task to establish some delay between learning of List A on the one side and its "short-delay" free recall and its category-cued recall on the other side. Moreover, the nonverbal testing is used as a distractor task, so the performance of long term memory can be tested by the following recall phases.

2.2.3 The Hopkins Verbal Learning Test

Another interesting memory test is the Hopkins Verbal Learning Test (HVLT) which is described in Brandt (1991) [17]:

> Each form of the Hopkins Verbal Learning Test (HVLT) consists of a 12-item word list, composed of four words from each of three semantic categories (see Appendix). The subject is instructed to listen carefully as the examiner reads the word list and attempt to memorize the words. The word list is then read to the subject at the approximate rate of one word every 2 s. The patient's free recall of the list is recorded. The same procedure is repeated for two more trials. After the third learning trial, the patient is read 24 words and is asked to say "yes" after each word that appeared on the recall list (12 targets) and "no" after each word that did not (12 distractors). Half of the distractors are drawn from the same semantic categories as the targets (related distractors) and half are drawn from other categories (unrelated distractors).

The word categories used in the HVLT are based on Battig & Montague (1969) [18], using the two most commonly given responses as semantically related distractor words to four other words chosen from this category. For example, for the category "four-legged animals", the two most commonly given responses are "dog" and "cat", so these responses are chosen as semantically related distractor words. Moreover, four target words are chosen from this category: "lion", "horse", "tiger" and "cow".

As the above-noted description of this test shows, two different retrieval strategies are used: free recall and recognition test. According to Brandt (1991) [17], two of the main advantages of the HVLT are its short duration and its retestability: The HVLT can be completed in about ten minutes, whereas other memory tests (e.g. the CVLT) take much longer time. Moreover, six equivalent forms of the HVLT are given (containing each different word sets), so subjects can be tested more than once with the HVLT without unintended learning effects.

2.3 Cognitive Workload

Workload is the amount of mental effort currently put on a human's mind. Hart & Staveland (1988) [1] define workload as

> a hypothetical construct that represents the cost incurred by a human operator to achieve a particular level of performance.

However, they admit that different individuals may define workload in different ways:

> The amount of "work" that is "loaded" on them, the time pressure under which a task is performed, the level of effort exerted, success in meeting task requirements, or the psychological and physiological consequences of the task represent the most typical definitions.

As these list of example definitions shows, workload is experienced differently by each individual, so it is hard to find an objective rating function or a general definition. However, in order to measure workload, it is necessary to find a common ground.

As Hart & Staveland state, workload can have different sources, e.g. time pressure, frustration and task difficulty. Some models of human cognition divide the process of cognition into different processing stages with each of them having access to different resources. One common view on cognitive systems divides the process of cognition into perception, cognition and response (see Craik, 1943, [19]). According to this division, the following types of workload can be distinguished:

- **Perceptional workload**: Workload of perception which can be further divided into visual workload, acoustical workload, etc. Perceptional workload depends e.g. on the number, rate and intensity of presented stimuli.
- **Cognitive workload**: Workload of internal mental thinking processes, depending e.g. on task complexity.
- **Physical workload**: Workload of physical response, depending e.g. on rate and complexity of physical response actions.

For the scope of this thesis, only cognitive workload is considered.

There are basically three ways of measuring a participant's cognitive workload:

- **Subjective criteria**:
  The participant is asked to report the perceived amount of workload. This can be supported by questionnaires especially designed for this purpose. One example is the NASA Task Load Index (TLX) developed by Hart & Staveland (1988) [1]: Participants are asked to report the perceived mental demand, the perceived physical demand, their frustration level etc. Participants have to provide a rating between 0 and 100 for each of the categories, and additional questions assemble a weighting of these categories. The TLX is then computed as weighted sum of the category scores. Using standardized tests to measure subjective criteria gives measures which are well comparable. However, this approach is very time-consuming.

- **Physiological correlates**:
  The experimenter measures physiological parameters and tries to conclude about the participant's workload state. Such physiological parameters are for example heart rate, skin conductivity or brain activity. So for example, at a high workload level the heart rate is usually higher than at a low workload level. However, a high heart rate does not directly imply a high workload level since other reasons might also lead to a high heart rate. Because of this, multiple physiological parameters should be measured to avoid false conclusions.

- **Performance**:
  As workload influences human behavior, performance parameters like error rate and response time also depend on the current workload. So by measuring these performance parameters, the experimenter can again make a guess about the participant's workload state. But again, a high error rate might be caused by high workload or for example by misunderstanding the task. So relying on performance parameters for measuring a participant's workload level also involves the thread of making false conclusions.

Of course, these three approaches can be combined, e.g. by measuring a participant's heart rate and skin conductivity in combination with an analysis of the participant's response time. By combining different approaches, the workload estimations should become more accurate because using different sources will stabilize the results. This holds since different approaches have different strengths and weaknesses (e.g. the probability of being influenced by some unrelated effect), and combining different approaches means combining their strengths and trying to eliminate their individual weaknesses.

2.4 Divided Attention

One way of increasing a participant's workload is the concept of divided attention: Instead of only executing one task, the participant is asked to execute two tasks concurrently. One of them is called the *primary task*, the other one the *secondary task*. Usually, performance in the primary task is of interest to the researcher, whereas the secondary task's only purpose is to increase the workload level. By varying the secondary task (e.g. in complexity or in time demands), the experimenter can vary the participant's workload level.

If the interest is only on cognitive workload, the two tasks should be designed in a way that there are no modality conflicts in perception and response. For example, if both tasks require the participant to read a word presented on a screen, this will result in higher perceptual workload than a scenario in which one tasks has visual input and the other one has auditory input (confer multiple resource theory by Wickens, 1984, [20]). When cognitive workload is measured, high perceptual or physical workload should be avoided due to their potential influence on the measured variables.

The focus of this section is on the paper of Craik et al. (1996) [3] that investigated similarities and differences between encoding and retrieval under divided attention.

Craik et al. conducted four experiments that contained learning and retrieval of words or word pairs. Encoding and retrieval were either performed under full attention or under divided attention (DA). Under DA, participants had to perform also a

secondary continuous reaction time (CRT) task. For the CRT task, four horizontally arranged boxes were presented on a screen. An asterisk appeared in one of these boxes and participants had to hit the corresponding key. When the correct key was pressed, the asterisk moved randomly to one of the other three boxes. The CRT task used visual stimuli for perception and manual responses, whereas the memory task used auditory stimuli and verbal responses. Between each subsequent encoding and retrieval phases, participants performed an arithmetic distractor task by adding 3 to auditory presented digits. This was done to eliminate effects of recency (which would be the participant repeating the words by using his/her ”inner voice”).

As Craik et al.’s results showed, DA at encoding time led to a marked drop in memory performance and only to a minor rise of response time (RT) in the CRT task. However, DA at retrieval had only a minor effect on memory performance, but caused a notable rise in RT of the CRT task.

Moreover, Craik et al. investigated the effects of task emphasis: Participants were either told to focus primarily on the memory task, to focus primarily on the CRT task, or to assign equal importance to both tasks. The results show that emphasis had a significant influence on RT for both encoding and retrieval: If the CRT task was emphasized, its RT was lower than in case of emphasis on both tasks or on the memory task. However, emphasis only affected memory performance under DA at encoding time with more emphasis on the memory task leading to better memory performance. Under DA at retrieval, memory performance under the three emphasis settings did not differ significantly.

Craik et al. also examined differences between the three retrieval techniques free recall, cued recall and recognition test. As they found out, free recall performance is impaired most by divided attention and performance in a recognition test is impaired least. Furthermore, DA at retrieval does not reduce memory performance at all if a recognition test is used as retrieval technique. These results are in line with their expectations since a recognition test is considered to provide most retrieval support (by representing the stimuli) in contrast to free recall which is considered to provide least retrieval support (none at all).

As the results from Craik et al. (1996) indicate, encoding is more vulnerable to DA than retrieval. This contrasts with the common opinion which considers encoding and retrieval to be very similar. Craik et al. state that this common opinion suggests

> that memory encoding processes consist essentially of operations whose primary functions are to perceive and understand external events, and that memory retrieval processes reflect efforts to reinstate the same pattern of mental and neural operations that existed at the time of the initial experience.

However, their results indicate that encoding requires attention and retrieval is a rather automatic process.

On the other hand, Fernandes & Moscovitch (2000) [21] report marked effects of DA at retrieval on memory performance if the secondary task is sufficiently similar to the primary task. For example, when the primary task consisted of learning and retrieving lists of words, the magnitude of effect of a word-based secondary task

was significantly larger than the magnitude of effect of a digit-based secondary task. However, this was only the case during retrieval – under DA at encoding time, both secondary tasks led to an impairment of memory performance of similar size. Fernandes & Moscovitch conclude that during retrieval, primary and secondary task compete for some common specialized resources (e.g. word-specific representational systems), whereas during encoding they compete mainly for general attentional resources.

There is a plethora of research in the field of divided attention which can not be covered in detail here. However, one interesting effect is reported by Logie et al. (2007) [22] who examined age related effects of DA: As their results show, DA at encoding time had more effect on memory performance of old participants than on memory performance of young participants. Moreover, the RT of old participants was significantly slower than the RT of young participants.

As the examples listed above show, divided attention is a practicable way of increasing a participant's workload, and moreover, it notably influences a participant's memory performance.

3. Experimental Design

In order to examine the influence of cognitive workload on declarative memory performance, an experiment was designed that used divided attention as a method for creating cognitive workload. Divided attention was used at encoding time and a recognition test was used as retrieval technique.

3.1 Method

A psychological experiment was conducted in which participants were asked to perform the following memory task:

A list of nouns was presented one by one on the screen at a pace of 1.5 seconds per word with a short break of 0.5 seconds (blank screen) between the presentation of two words. The participants were told to memorize these words.

As a distractor task, participants had to count backwards in threes from a random three-digit number presented on the screen (e.g. with 328 presented, participants had to count: 328, 325, 322, 319, ...). This distractor task was adopted from Fernandes & Moscovitch (2000) [21]. After 20 seconds, participants were disrupted by an audio signal. Participants were told to count as far as possible without being inaccurate. Since the single purpose of this task was to eliminate recency, no recordings were made.

In the following recognition test, a list of eight nouns was presented on the screen one by one and the participants had to decide for each word whether they had learned it before. If they remembered the word, they had to press "Y", otherwise they had to press "N". Participants were told to respond as fast and as accurately as possible. After pressing "Y" or "N", the next word appeared on the screen. Both response time and response correctness were recorded. Half of the presented words were target words, which had been learned before, the other half were distractor words. Participants were not told how many target words there would be in the recognition test. Each word (both targets and distractors) was retrieved only once during the whole experiment to avoid unintended learning effects.

These three steps (learning – distractor task – retrieval) were repeated six times, forming a block. Before each learning and retrieval phase, a short informational

message was presented for five seconds. After each retrieval phase within a block, there was a break of ten seconds.

Each block used a word list containing 54 words (24 target words, 24 distractor words and 6 filler words that were learned but never retrieved). The first four learning phases of a block consisted of eight words, the last two learning phases of a block consisted of five words. There were no retrievals across blocks, and participants were told so. All words used in this experiment were German nouns with a mean length of 5.82 letters (SD: 2.99, range: 2-12) and a mean number of syllables of 1.89 (SD: 0.48, range: 1-4). See appendix A.1 for a list of all words used in this experiment.

Half of the 24 words learned during a block were presented for learning only once, half of the words were presented in two subsequent learning phases ("reinforcement" property). Half of the words learned during a block were presented in the retrieval phase right after they had been learned, half of the words were presented with a delay of one retrieval phase ("gap" property). Half of the 48 words retrieved in each block were so called "cluster words": There were four clusters per block and each cluster consisted of six semantically related words (e.g. cat, dog, cow, horse, lion, tiger). Half of the words of each cluster were used as target words, half of them were used as distractor words. The clusters and their words were taken from Battig & Montague (1969) [18].

The three dimensions "cluster" (cluster words vs. singleton words), "reinforcement" (word learned once or twice) and "gap" (direct retrieval vs. delayed retrieval) were counterbalanced, so that all cells (e.g. cluster words that were reinforced twice and retrieved without a gap) were of equal size. Assignment of words to the "cluster" property was given a priori. For each participant, words were assigned randomly to the property "target" (word to learn or distractor word), and in case of a target word they were assigned again randomly to the properties "reinforcement" and "gap". See appendix A.2 for a detailed description of a block's structure.

To induce different workload levels, two secondary tasks based on the "switching task" were used, similar to those mentioned in Monsell (2003) [23]:

In the "easy digit task", a randomly chosen sound file, containing a digit, was played concurrently with each word appearing on the screen during a learning phase. Participants were asked to report verbally for each digit, whether it was "large" or "small". A digit was considered to be "large", if it was greater than or equal to 5, and "small" otherwise. Participants were asked to report their answer as accurately as possible before the next word was presented on the screen and the next sound file was played. Participants were told that the memory task and the easy digit task were of equal importance.

In the "difficult digit task", again a randomly chosen sound file, containing a digit, was played concurrently with each word appearing on the screen during a learning phase. Participants were asked to report verbally alternating whether the digit was "large" or "small" and whether the digit was "odd" or "even". They were asked to start with the "large/small" decision. For example for the digit sequence "three", "eight", "nine", "one", the correct responses would have been "small", "even", "large", "odd". Again, participants were asked to report their answer as accurately as possible before the next word was presented on the screen and the next sound file was played.

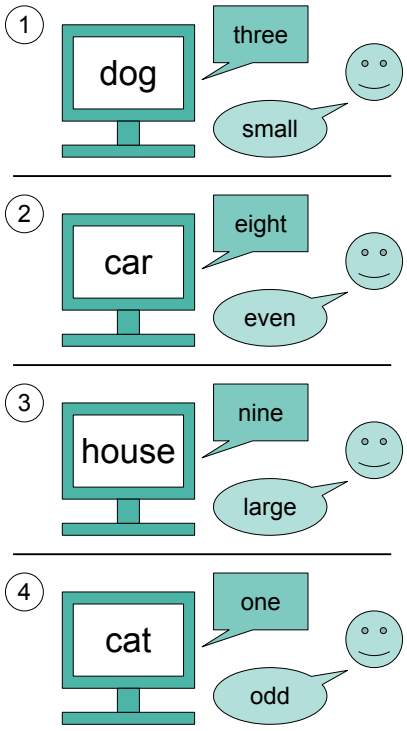

Figure 3.1: This figure illustrates the concurrent performance of both the memory tasks (words to memorize presented on the screen) and the difficult digit task (digits to classify presented auditory by playing sound files).

Participants were told that the memory task and the difficult digit task were of equal importance.

Figure 3.1 illustrates how difficult digit task and memory task are performed concurrently.

Both secondary tasks were designed to avoid modality conflicts with the memory task in perception and response: In the memory task, stimuli were presented visually and responses were made manually, whereas in both digit tasks, stimuli were presented auditory and responses were made verbally. Since the memory task operated on words, both secondary tasks were designed to operate mainly on digits to avoid conflicts in verbal memory. However, since both memory task and secondary tasks require cognitive resources, cognitive workload should arise. An audio recording was made in order to analyze the verbal responses in the secondary task.

In this experiment, three workload modes can be distinguished:

- In workload mode None, participants performed solely the memory task.
- In workload mode Low, participants performed the memory task and the easy digit task concurrently with equal importance assigned to both tasks.
- In workload mode High, participants performed the memory task and the difficult digit task concurrently with equal importance assigned to both tasks.

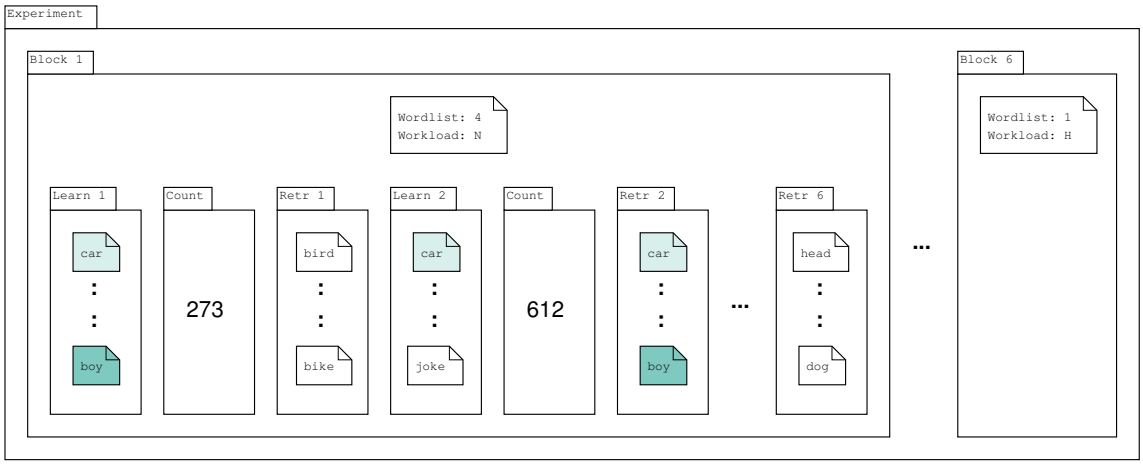

Figure 3.2: Overall experiment structure (training blocks omitted for simplicity). In this example, "car" is a cluster word that is reinforced twice and retrieved without a gap (another word of the same cluster is "bike"). "boy" is a singleton word that is reinforced once and retrieved with a gap. When denoting workload modes, N stands for workload mode None and H stands for workload mode High.

There were nine blocks in total: three training blocks at the beginning (one of each workload mode) and six test blocks (two of each workload mode). Only data from the test blocks was analyzed. Between two subsequent test blocks there was a break of 90 seconds.

Training blocks were shorter than test blocks: They contained four learning and retrieval phases. Three of the learning phases consisted of five words and the last one consisted of three words. All retrieval phases contained five words, three of them target words and two of them distractor words. Breaks between two subsequent training blocks were 60 seconds long.

Figure 3.2 shows the overall structure of the experiment (with training blocks omitted for simplicity).

Order of underlying word lists and order of workload modes were counterbalanced across participants for test blocks. The order of workload modes was constrained by the following two restrictions: Two subsequent blocks had to be of different workload modes and both halves of the test blocks had to cover all three workload modes. See appendix A.3 for more information on the counterbalanced design. The training blocks were identical for all participants.

Participants were not allowed to speak during the experiment, except when performing the distractor task or when performing the easy or the difficult digit task. Speaking was also allowed in breaks between blocks.

The experiment itself took about 65 minutes and was conducted with 24 participants (mean age: 20.75 years, SD: 3.37, range: 15 – 29). 17 of them were male (mean age: 20.24 years, SD: 2.88, range: 15 – 28), 7 of them were female (mean age: 22.00, SD: 4.32, range: 16 – 29). A majority of 17 of them were students, 4 of them were trainees and 3 were young professionals. Per participant, for each cell (e.g. cluster words that were reinforced twice and retrieved without a gap in workload mode None) a total number of 6 data points was recorded.

The experiment was run on a MacBook Pro 13" (Intel Core 2 Duo 2.26 GHz, 2 GB RAM) using the PsychoPy framework (see Peirce, 2007, [24] and Peirce, 2009, [25]). See appendix B.1 for more information on the scripts used for conducting this experiment.

3.2 Results

The main questions to be answered by the results of this experiment are the following:

- Is there a general impact of workload on human memory performance?
- Are there effects of learning by repetition and effects of decay?
- Can any differences be observed between semantically related and semantically unrelated words?
- Can the arising effects be observed in all workload modes?
- Do the aspects listed above interact in some way or are they independent?
- Is there a relationship between response time and response quality?

The data gathered during the experiment was analyzed using the statistical toolkit R. Three dependent variables were examined: response time, hit rate and false alarm rate. Moreover, accuracy was computed from hit rate and false alarm rate and was also analyzed.

- *Response time* (rt) was measured as time difference between the presentation of a word and the participant's keystroke, and is given in seconds.
- *Hit rate* (hit) was calculated on target words as fraction of correct responses, and is given as fraction.
- *False alarm rate* (fa) was calculated on distractor words as fraction of incorrect responses, and is given as fraction.
- *Accuracy* (acc) was defined as "hit rate - false alarm rate" (as in Craik et al., 1996, [3]), and is also given as fraction. Note that the expected accuracy of random guessing is zero.

The following dimensions were analyzed:

- *Reinforcement*: Was the word presented for learning once or twice?
- *Gap*: Was the word retrieved directly after the last reinforcement or was it retrieved with a gap?
- *Cluster*: Was the word a cluster word or a singleton word?
- *Target*: Was the word a target word or a distractor word?

- *Correctness of Response*: Was the participant's response correct or not?
- *Age*: Was the participant older than 20 years or not?
- *Gender*: Was the participant male or female?
- *Block*: In which block was the word learned and retrieved?
- *Word List*: To which word list did the word belong?

The results regarding the following dimensions are not included in this section:

- **Age**: All participants were between 15 and 29 years old. Due to this limited range of age, the results obtained from this dimension can hardly be generalized. Moreover, the participants' age was not equally distributed in this range. Since the observed effects are therefore not reliable, they were neglected for the purpose of this thesis.
- **Gender**: Since 17 participants were male and only 7 participants were female, the results from analyzing the gender dimension are not reliable due to the different sizes of the two groups. So also the results for analyzing the gender condition were neglected for the purpose of this study.
- **Block**: Due to the counterbalanced experiment design, any effect of block will leave all other dimensions unaffected, so also this dimension was omitted. Moreover, modeling possible effects of fatigue that could be observed by analyzing the block dimension would be beyond the scope of this thesis.
- **Word List**: Again, due to the counterbalanced experiment design, any effect of word list will leave all other dimensions unaffected. Since analyzing the word list dimension will not lead to any insights regarding the aim of this thesis, also the word list dimension was omitted.

Before analyzing the dependent variables listed above, one general observation was made when analyzing performance in the secondary tasks: Participants made more errors in the secondary task in workload mode HIGH (mean: 2.680, SD: 2.519) than in workload mode LOW (mean: 0.125, SD: 0.393). A two-sided t-test ($\alpha = 0.05$) showed that this difference was significant.

Regarding the different dependent variables, the following dimensions were analyzed:

- **Response time**: reinforcement, gap, cluster, target, correctness of response
- **Hit rate**: reinforcement, gap, cluster
- **False alarm rate**: cluster
- **Accuracy**: cluster

Effects of Workload			
Dependent Variable	None	Low	High
Response Time	0.92701 (0.44917)	1.01688 (0.56657)	1.09282 (0.68536)
Hit Rate	0.88115 (0.29095)	0.69922 (0.41338)	0.59478 (0.43863)
False Alarm Rate	0.04549 (0.11950)	0.09722 (0.17528)	0.14306 (0.21808)
Accuracy	0.83611 (0.23246)	0.60972 (0.30167)	0.45799 (0.35273)

Table 3.1: This table shows the mean values and the standard deviations of the dependent variables across the workload modes. Each cell contains "mean (SD)" for the specified dependent variable in the specified workload mode. Response Time is given in seconds. Hit rate, false alarm rate and accuracy are given as fraction.

Significance was always tested by using a one-way or two-way ANOVA with a significance level of $\alpha = 0.05$.

As a one-way ANOVA showed, there was a significant effect of workload regarding all four dependent variables. Table 3.1 lists the means and standard deviations across workload modes and figure 3.3 illustrates these results by using bar charts. As it can be easily seen, hit rate and accuracy decrease with increasing workload, whereas response time and false alarm rate rise with increasing workload. Note that it is response time at *retrieval* that is influenced by workload at *encoding* time.

When focusing on response time, the following main effects were observed:

- **Effect of reinforcement**:
  Response time of words reinforced twice is significantly lower than response time of words reinforced only once. This effect can be observed when doing an overall analysis considering data from all workload modes, and when considering only workload mode None. In workload modes Low and High, this relationship still holds for the mean values, but the effect is not significant.

- **Effect of gap**:
  Response time of words retrieved directly after their last reinforcement was significantly lower than response time of words retrieved with a gap after their last reinforcement. This effect can be observed in workload modes None and Low, and when doing an overall analysis. In workload mode High, however, this effect is not significant.

- **Effect of target**:
  Response time of target words was significantly lower than response time of distractor words. Again, this effect can be observed in workload modes None and Low, and in an overall analysis, but not in workload mode High.

- **Effect of correctness**:
  Response time of correct responses was significantly lower than response time of incorrect responses. This effect can be observed in all of the three workload modes None, Low and High, and in an overall analysis.

- **Effect of cluster for distractor words:**
  When only analyzing distractor words, response time of cluster words was

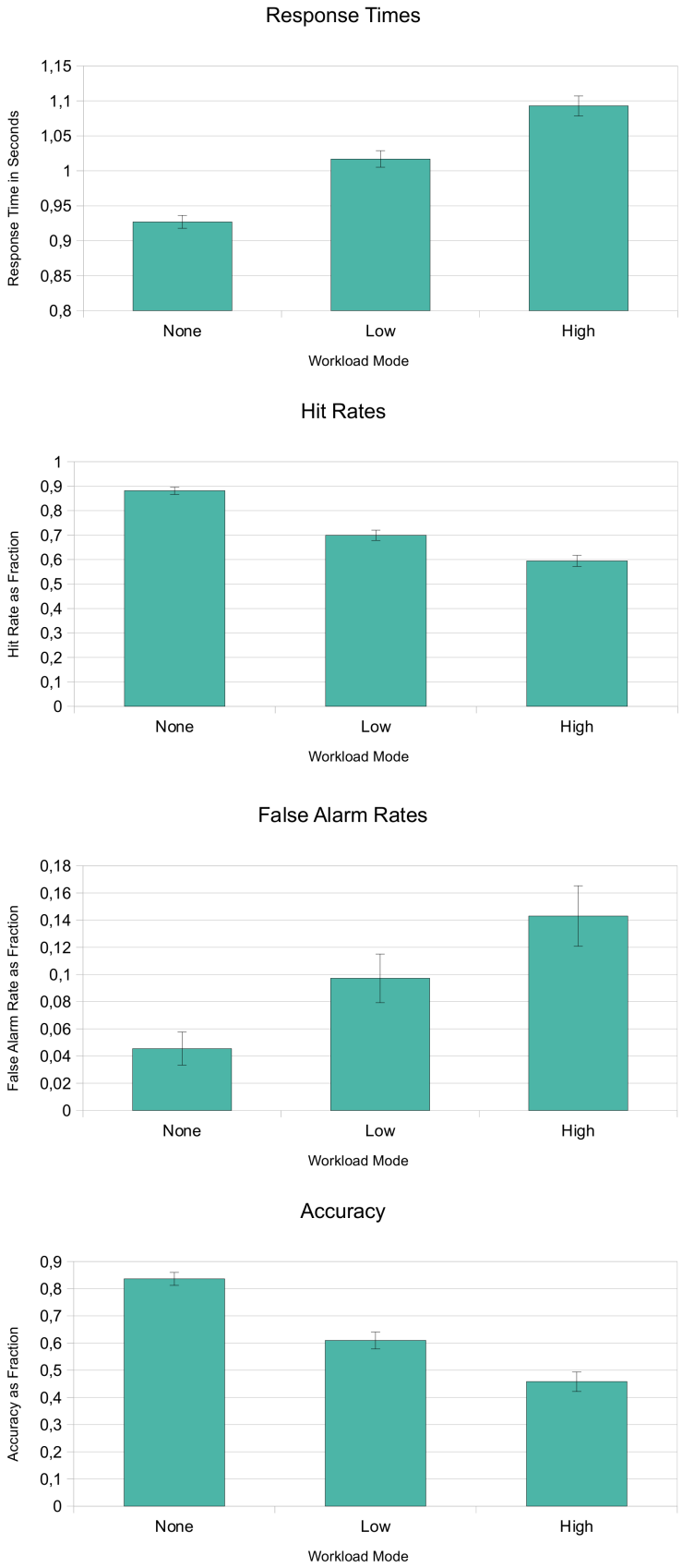

Figure 3.3: This figure illustrates the contents of table 3.1 by using bar charts. Bars represent mean values of the given workload mode and error bars represent the corresponding standard error.

significantly higher than response time of singleton words. This effect was observed in workload modes NONE and HIGH, and in an overall analysis. In workload mode LOW, this effect was not significant.

- **Effect of cluster**:
  A general effect of cluster was observed in workload mode NONE: Response time of cluster words was significantly higher than response time of singleton words. However, no such general cluster effect could be observed for the remaining workload modes LOW and HIGH. Also in an overall analysis, this effect did not arise.

Note that the number of observed effects regarding the response time decreases from six significant effects in workload mode NONE, over three significant effects in workload mode LOW, to only two observed effects in workload mode HIGH.

However, the analyzed response times contain also the response times of guessing – which might bias the results. Because of this, response time was also analyzed when considering only correct answers which were considered to be less biased by guessing. The following effects were observed:

- **Effect of reinforcement**:
  Again, in workload mode NONE and in an overall analysis, response time of words reinforced twice was significantly lower than response time of words reinforced only once.

- **Effect of gap**:
  In all workload modes (NONE, LOW, HIGH) and in an overall analysis, there was a significant effect of gap. Words retrieved without a gap had a significantly lower response time than words retrieved with a gap.

- **Effect of target**:
  In all workload modes and in an overall analysis, response time of target words was significantly lower than response time of distractor words.

- **Effect of cluster for distractor words**:
  The effect of cluster distractor words having a significantly higher response time than singleton distractor words was observed only in workload mode NONE and in an overall analysis.

- **Effect of cluster**:
  Again, only in workload mode NONE a general effect of cluster could be observed with cluster words having a significantly higher response time than singleton words.

When focusing on hit rate, the following main effects were observed:

- **Effect of reinforcement**:
  In all workload modes (NONE, LOW, HIGH), as well as in an overall analysis, hit rate of words reinforced twice was significantly higher than hit rate of words reinforced only once.

- **Effect of gap**:
  Hit rate of words retrieved directly after their last reinforcement was significantly higher than hit rate of words retrieved with a gap between their last reinforcement and their retrieval. This effect was observed in all workload modes, as well as in an overall analysis.

- **Effect of cluster**:
  In workload modes LOW and HIGH, and in an overall analysis, hit rate of cluster words was significantly higher than hit rate of singleton words. However, in workload mode NONE, this effect was not significant.

When focusing on false alarm rate, one main effect was observed:

- **Effect of cluster**:
  Cluster words had a significantly higher false alarm rate than singleton words in workload modes NONE and HIGH, as well as in an overall analysis. In workload mode LOW, however, this effect was not significant.

When focusing on accuracy, no further significant effects could be observed, apart from the effect of declining accuracy for increasing workload.

Moreover, the different clusters were compared to find cluster dependent effects (e.g. one cluster having significantly higher hit rate than all other clusters), but no significant effect could be observed.

See appendix A.4 for a table containing mean values and standard deviations for all effects listed above.

By doing several two-way ANOVAs (one for each dependent variable), no significant interaction effects regarding the analyzed dimensions could be observed. All observed interaction effects contained at least one of the dimensions block, word list, age or gender. However, as described above, these dimensions were omitted from analysis.

Figures 3.4 and 3.5 show four interaction plots: response time as a function of reinforcement and cluster, response time as a function of reinforcement and gap, hit rate as a function of reinforcement and cluster, and hit rate as a function of reinforcement and gap. The approximately parallel curves indicate that there is no interaction between reinforcement and gap, and reinforcement and cluster, regarding both hit rate and response time.

Figure 3.6 shows response time for correct answers as a function of workload mode and target. Although no significant interaction effect could be observed, in each of the three workload modes a main effect of target exists. When looking at figure 3.6, two further observations can be made: Each curve (for target and distractor words) is approximately on a straight line. Moreover, the difference between both curves in workload mode NONE is nearly twice as large as it is in workload mode HIGH (as indicated by the bars).

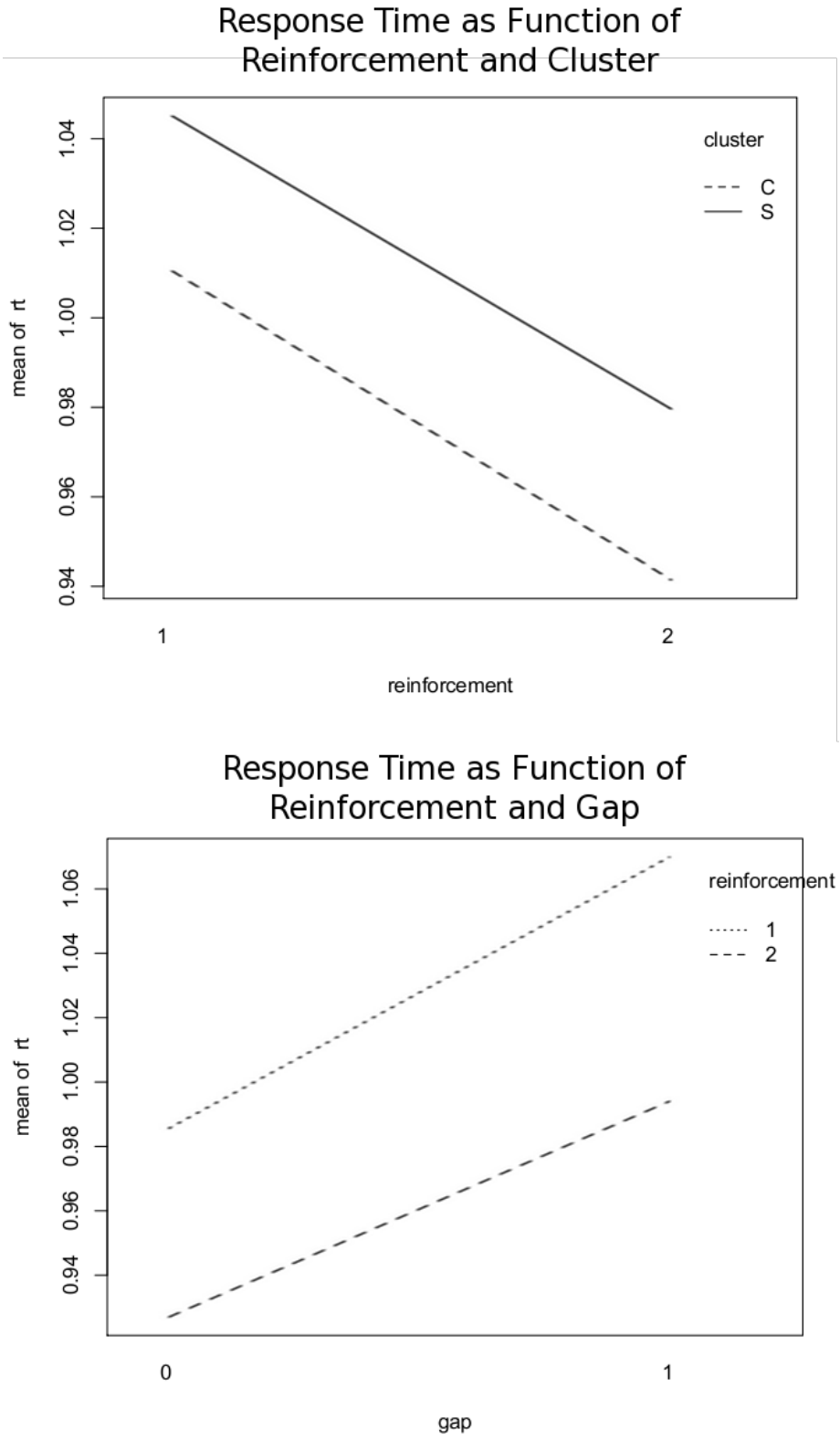

Figure 3.4: This figure shows two interaction plots illustrating response time as function of reinforcement and cluster, and as function of reinforcement and gap, respectively. ”C” stands for cluster words and ”S” for singleton words. Gap ”0” corresponds to words retrieved without a gap and gap ”1” to words retrieved with a gap.

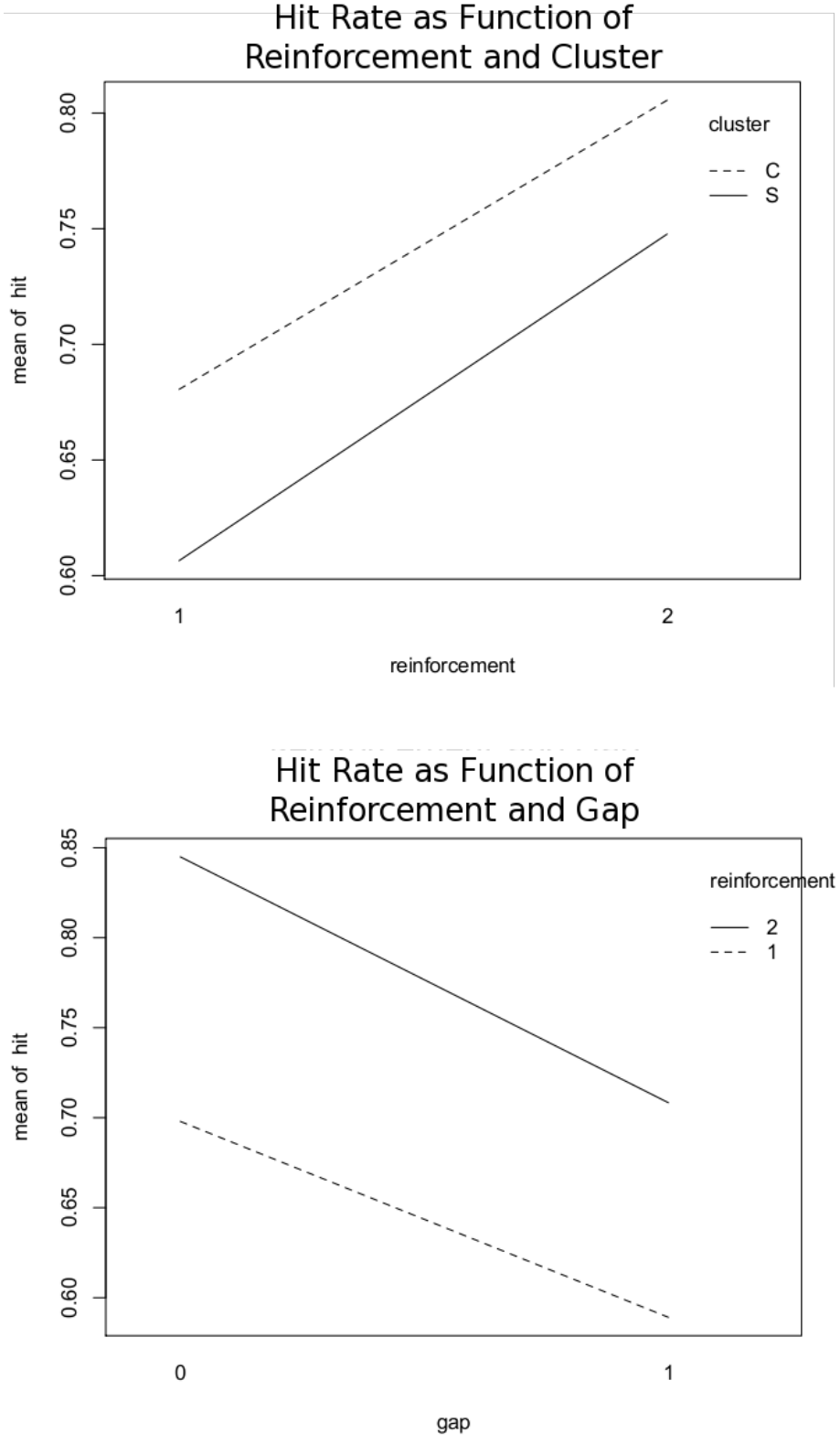

Figure 3.5: This figure shows two interaction plots illustrating hit rate as function of reinforcement and cluster, and as function of reinforcement and gap, respectively. ”C” stands for cluster words and ”S” for singleton words. Gap ”0” corresponds to words retrieved without a gap and gap ”1” to words retrieved with a gap.

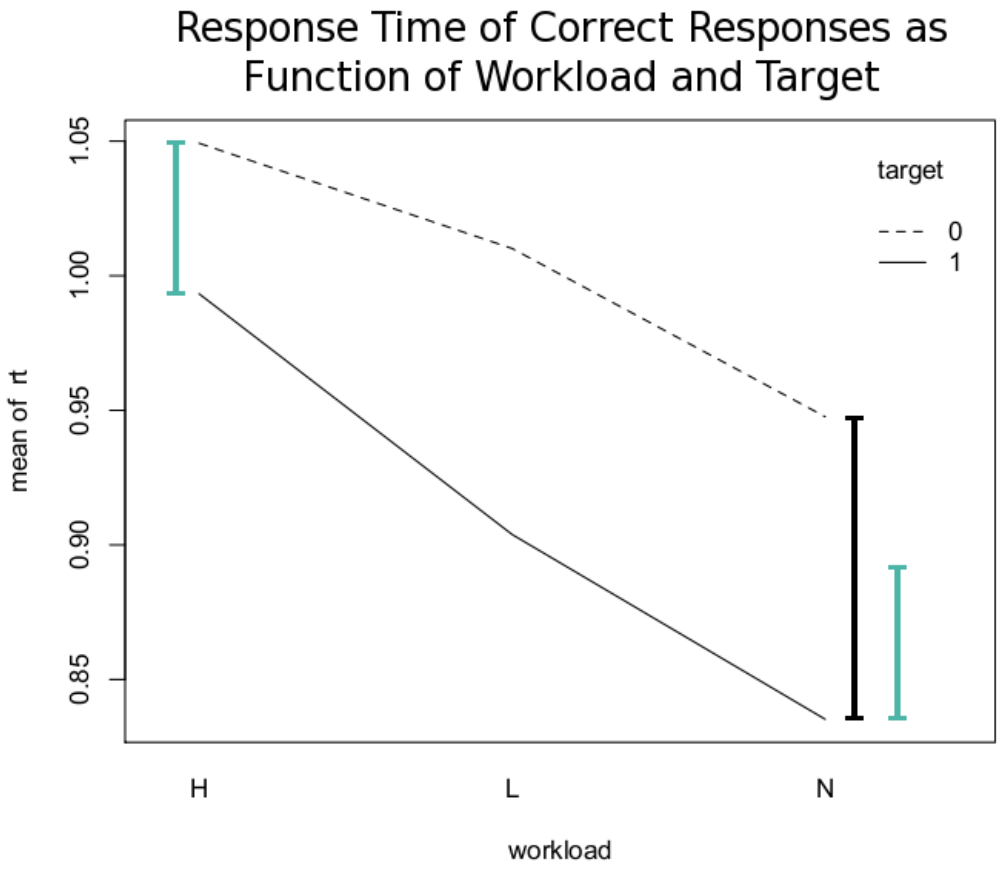

Figure 3.6: This interaction plot shows response time for correct answers as a function of workload mode (x-axis) and target (solid vs. dashed line). The solid line represents mean response time for correctly recognized target words, whereas the dashed line represents mean response time for correctly recognized distractor words. On the x-axis, "H" stands for workload mode HIGH, "L" for workload mode LOW and "N" for workload mode NONE.

3.3 Discussion

The results from this experiment prove that workload has an impact on human memory performance: Not only quality of retrieval (measured as hit rate, false alarm rate and accuracy) is affected by workload, but also the time needed for retrieval (measured as response time). Both the worse performance under high workload and the higher amount of errors made in the secondary task under high workload approve that workload mode HIGH was more demanding than workload mode LOW and workload mode NONE. Performance decreases in both primary and secondary task as workload gets higher, which indicates that participants did not give up on one of the tasks to fully concentrate on the other one.
One possible explanation for this observed effect is the Shared Time Model proposed by Craik et al. (1996) [3] which describes memory performance as a function of available encoding time: As the secondary task gets more complicated, less time is available for encoding and this leads to a rather shallow encoding in memory. Retrieval of a word being less deeply encoded is then having a lower probability of success than retrieval of a deeply encoded word.
The interesting fact of response time at *retrieval* being influenced by workload at *encoding* time suggests that also retrieval time is dependent on the depth of encoding.

Effects observed for reinforcement and gap indicate learning by repetition and forgetting: When learned twice, a word can be retrieved faster and with greater probability than in case it was only learned once, so repetitive learning seems to lead to deeper encoding. Furthermore, a gap in retrieval leads to higher response time and to lower retrieval probability, which can be easily explained by the effect of forgetting information encoded further away in the past. However, this effect of decay over time is not uncontroversial in the psychological community: For example, Lewandowsky & Oberauer (2009) [26] argue that there is no time-dependent decay. They propose a model where retrieval probability is dependent on the time available for restoring the memory. When this restoration time is reduced due to other activities (e.g. post-error processing or some other memorizing processes), the probability of successful retrieval declines. This is also a reasonable explanation since if a word is retrieved with a gap, not only the temporal distance between encoding and retrieval, but also the number of words learned and retrieved between encoding and retrieval of this word is larger than for a word retrieved without a gap. However, since the model devised to reproduce this effect is based on the ACT-R declarative module which assumes decay to be time-dependent, the theory by Lewandowsky & Oberauer will not be considered for the remainder of this thesis.

The effect of cluster on hit rate with cluster words having a higher hit rate than singleton words indicates that it is easier to learn semantically connected chunks of information than to learn semantically unrelated words. However, since this effect was not observed for workload mode NONE, it might be that this chunking strategy is only adopted when limited resources are available for encoding. Of course, it is also possible that the effect does also exist for workload mode NONE, but that its effect size was too small to be measured in this experiment. Nevertheless, when considering response times, there is only an effect of cluster for workload mode NONE – with cluster words having a higher response time than singleton words.
This is a rather unexpected result: If a high hit rate can be attributed to deep encoding and a short response time can also be attributed to deep encoding, one

would probably expect the response time of cluster words to be significantly lower than the response time of singleton words. Since for calculating the hit rate, only target words are considered, at least a response time effect of cluster for target words might be expected. However, no such effect could be observed. Again, it is possible that this effect exists, but was not uncovered by this experiment. Moreover, since Craik et al. (1996) [3] did not address the issue of semantically related words, their Shared Time Model might only be valid for singleton words.

When considering false alarms (i.e. distractor words erroneously being recognized as target words), cluster words have a significantly higher false alarm rate than singleton words. This can be interpreted as participants mixing up semantically related nouns: When a participant has encoded the words "emerald", "diamond" and "opal", he/she might memorize them as "some gems". Being asked to retrieve "sapphire", he/she might then erroneously recognize this word since it belongs to the same category. As the effect of cluster on distractor words' response time suggests, it seems to be a more difficult task to decide for a cluster word whether it has been encoded before than for a singleton word. This is in line with the higher false alarm rate mentioned before: As making a decision for a cluster distractor word is more difficult than making a decision for a singleton distractor word, the process of making this decision takes more time and leads to more errors.

The results regarding cluster words vs. singleton words are contrary: On the one hand, cluster words have a significantly higher hit rate than singleton words which suggests that cluster words can be encoded better than singleton words due to their semantical relationship to other words. On the other hand, cluster words also have a significantly higher false alarm rate than singleton words which suggests that cluster words are also easier to confuse with other semantical related words. This would mean that having semantical relationships to other words is both beneficial and adverse for correct retrieval. Therefore, the effects of cluster words vs. singleton words require further research. However, these contrary effects can be easily modeled by using a spreading approach as in LTM^C: cluster words receive more spreading activation than singleton words (since they have more connections to context-relevant words) and a higher spreading activation will result in a higher probability of retrieval – which leads both to a higher hit rate (probability of retrieval of target words) and a higher false alarm rate (probability of retrieval of distractor words).

The observation that there are no significant effects of accuracy can be explained by this contrary results for the cluster dimension: Since accuracy is computed as difference between hit rate and false alarm rate, and since only the cluster condition is considered, the observed contrary cluster effects neutralize. As cluster words have both a higher hit rate and a higher false alarm rate than singleton words, their accuracy is not significantly higher than the accuracy for singleton words because both effects get subtracted.

When comparing the results of analyzing response time and of analyzing response time of correct responses, only minor differences can be found: When analyzing response time of correct responses, the effects of gap and target are also significant in workload mode HIGH, but the cluster effect when only considering distractor words is not significant in workload mode HIGH.

The significant effect of target on response time indicates that identification of an encoded word is faster than identification of a previously unseen word. Moreover,

correct responses being faster than incorrect ones suggests that participants responded fast when they were sure and slowly when they were unsure and needed some time for consideration. This additional time of consideration could also be an explanation for the higher response time of distractor words. However, these effects on response time require further research.

The approximately parallel curves in figures 3.4 and 3.5 indicate that the effects of reinforcement and cluster are independent of each other. This also seems to apply to the effects of reinforcement and gap. The consequence of this observation is that for devising a model to mimic the results of this experiment, these effects can be modeled independently of each other.

Although there was no significant interaction effect observed regarding response time, the number of significant effects within each workload mode indicate that the differences in response time converge with increasing workload. Figure 3.6 illustrates this effect: Although no significant interaction effect was observed, the differences in response time between target words and distractor words are approximately halved when considering workload mode HIGH compared to workload mode NONE. So although there is no significant effect, a slight tendency can be observed.

3.4 Conclusions

This experiment confirmed that there are effects of learning by practice and effects of forgetting. Thus, it indirectly confirmed the plausibility of the ACT-R equation for the base level activation which takes into account exactly these effects. Hence, the ACT-R declarative module seems to be a good starting point for modeling the observed effects. Moreover, the Shared Time Model by Craik et al. (1996) [3] was confirmed at least in parts. However, there is more research necessary, e.g. regarding the effects of cluster words. One potential approach of modeling the observed cluster effects is the spreading approach used in LTM^C. The observation of some effects being independent of each other can be taken into account by modeling these effects independently of each other. In the following chapter, a model is devised for predicting the effects observed in this experiment.

4. Design

As stated in the introduction, the purpose of this bachelor thesis is not only to conduct a psychological experiment and to analyze the results of this experiment, but also to devise a model which can reproduce the observed effects. The described WorkloadMemoryModel (WMM) will only try to reproduce the observed mean values, but not the variances. First, the general model design is described and key design decisions are discussed. Second, the implementation of the model is outlined.

4.1 General Model Design

This section describes the general design of the WorkloadMemoryModel and discusses key design decisions.

4.1.1 Design Description

The WorkloadMemoryModel (WMM) devised in this thesis brings together the concepts of the ACT-R declarative module and of LTM^C with the WordNet database. Note that there has already been research in adapting the ACT-R cognitive architecture to different workload modes (see Pröpper & Putze, 2011, [27]). However, their research has focused on the overall architecture's performance under workload, whereas this thesis specifically addresses the memory model.

The basic element of the WMM is called a *memory element*. Every memory element corresponds to a WordNet noun synset, has a unique identifier (the WordNet synset ID) and is connected to other memory elements via a parent/child relationship. This relationship is obtained by the hypernym/hyponym relation between WordNet noun synsets.

Probability of retrieval and retrieval time depend like in ACT-R on an activation value. This activation value consists of base level activation and spreading activation.

The base level activation is computed according to the ACT-R equation:

$$B(i) = \ln(\sum_{k=1}^{n} (t_c - t_k)^{-d}) \tag{4.1}$$

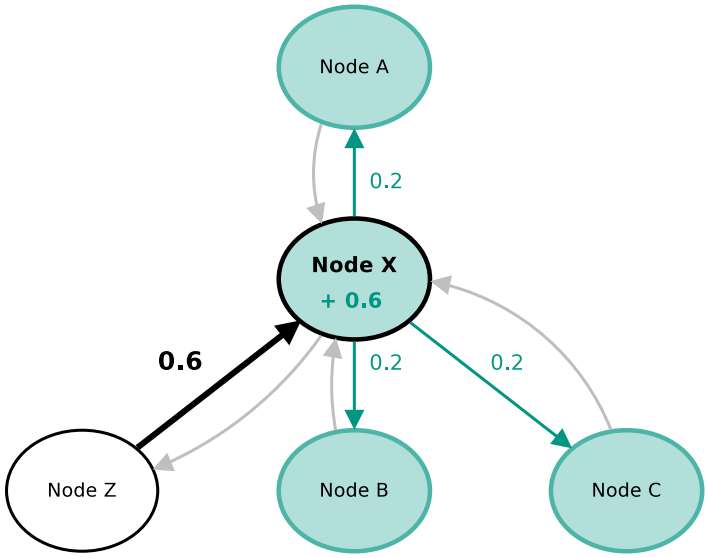

Figure 4.1: This figure illustrates the spreading step. An activation value of $s_{in} = 0.6$ is spread from node Z to node X. Node X divides this amount of spreading activation by the number of neighbors that have not been visited yet (which is $m = 3$) and spreads the new value $s_{out} = 0.6/3 = 0.2$ on to its neighbors. Moreover, node X increases its local spreading activation by $s_{in} = 0.6$. The grey edges represent hypernym/hyponym pointers that are not used in this spreading step.

The free parameter d is called the *decay parameter*. The ACT-R equation leaves also the unit of time as a degree of freedom for the modeler. For the WMM, times t_k and t_c will be measured in minutes since the experiment start.

To provide all memory elements with a reasonable initial activation value, all elements are assumed to be encoded at time $t = -60$ min. This prevents the activation value from becoming negative infinity.

As a simplification, all encodings within a learning phase and all retrievals within a retrieval phase are assumed to take place at the same point in time, respectively, which is defined as the center of the respective phase. The time difference between a learning and a subsequent retrieval phase is set to 37 seconds, and the time difference between a retrieval and a subsequent learning phase is set to 27 seconds for the conducted experiment.

The spreading activation is computed in a similar fashion as in LTM^C, but with base level activations as starting point instead of external stimulation: First, at each time step, the base level activation of each memory element is computed.

Then, each of the memory elements that have a value high enough to be spread is selected as a starting point for spreading. Spreading itself works in a breadth first search manner: Starting at a memory element i, the received spreading activation s_{in} (either spread to i by some of its neighbors, or the starting value if i is the current starting point for spreading) is divided by the number m of neighbors that have not been visited, yet. If the resulting value $s_{out} = s_{in}/m$ is greater than a specific *spreading threshold* τ_{spread}, it is spread to the neighboring nodes in a BFS manner. If i is not the current starting point of spreading, s_{in} is added to its local

spreading activation. Figure 4.1 illustrates this spreading step. After processing i, the next element is chosen according to the BFS order and processed in the same way. This is done until the spreading value falls below τ_{spread} everywhere.

Since base level activations can be negative, but only positive values should be spread through the memory element network, a *spreading potential* P_{spread} is added to the base level activation and this sum is multiplied with the factor 10. If the resulting value $10 \cdot (B(i) + P_{spread})$ is higher than τ_{spread}, it will be spread amongst the memory element network. To reduce the number of free parameters, like in LTM^C, the spreading threshold is set in dependence of the starting value. In this case, it is set according to equation 4.2 at the beginning of each spreading procedure:

$$\tau_{spread} = \left|10^{-4} \cdot (10 \cdot (B(i) + P_{spread}))\right| = \left|10^{-3} \cdot (B(i) + P_{spread})\right| \tag{4.2}$$

This procedure of spreading is run for every memory element as a starting point. After that, every node has received a total amount $S(i)$ of spreading activation. The overall activation of this node is then computed by adding its base level activation and its spreading activation as in equation 4.3:

$$A(i) = B(i) + S(i) \tag{4.3}$$

In contrast to ACT-R and LTM^C, no noise activation is added.

As in ACT-R, the probability of retrieval can be computed according to equation 4.4, where τ is the *retrieval threshold* and s is a factor reflecting the *retrieval sensitivity*.

$$P(i) = \frac{1}{1 + e^{-(A(i)-\tau)/s}} \tag{4.4}$$

When computing the response time for a retrieval request, the WMM makes a distinction between elements that have been retrieved successfully (equation 4.5) and elements that have not been retrieved (equation 4.6). In both cases, I is the *intercept time* needed for visual perception and manual response and F is the *retrieval latency factor*.

$$rt_{retrieved}(i) = I + F \cdot e^{-A(i)} \tag{4.5}$$

$$rt_{notRetrieved}(i) = I + F \cdot e^{\tau + f \cdot A(i)} \tag{4.6}$$

According to the relationship between F and τ mentioned in Anderson et al. (2004) [5] (see also section 2.1.1), in the WMM, F is always set to $0.35 \cdot e^{\tau}$. In case the element is not retrieved from memory, there is an additional parameter f which is called the *distractor factor*. It determines the influence of a memory element's activation on the response time in case it can not be retrieved. The distractor factor f is not allowed to be negative and aims to model the effect of cluster distractors having a higher response time than singleton distractors since in the WMM they receive a higher amount of spreading activation.

For each retrieval request, the WMM will return two values: The retrieval probability $P(i)$ and the expected response time $rt_{expected}(i)$ which can be computed as in equation 4.7:

$$rt_{expected}(i) = P(i) \cdot rt_{retrieved}(i) + (1 - P(i)) \cdot rt_{notRetrieved}(i) \tag{4.7}$$

In total, the model has eight parameters: decay d, intercept time I, retrieval threshold τ, latency factor F, spreading potential P_{spread}, spreading threshold τ_{spread}, distractor factor f and retrieval sensitivity s. Since F is computed dependent on τ, and τ_{spread} is computed dependent on P_{spread} and $B(i)$, six free parameters remain.

The design as stated until now does not contain any component for modeling different workload modes. This is simply done by using three different parameter sets – one for each workload mode. So for each word in workload mode NONE, the NONE parameters are used, for each word in workload mode LOW, the LOW parameters are used, and for each word in workload mode HIGH, the HIGH parameters are used.

For calculating the base level activation $B(i)$ of a memory element, the three workload parameter sets might interact: For each summand $(t_c - t_k)^{-d}$ in equation 4.1, the decay parameter of the workload parameter set corresponding to encoding time t_k is used. For example, if a memory element is encoded at t_0 when the workload mode was NONE, and at t_1 when the workload mode was LOW, then the base level activation is computed as $B(i) = \ln((t_c - t_0)^{-d_{None}} + (t_c - t_1)^{-d_{Low}})$ with d_{None} and d_{Low} being the decay parameters of parameter set NONE and LOW, respectively. This is the only interaction happening between the three workload modes – spreading activation, retrieval probability and response time are each calculated independently of the other parameter sets.

4.1.2 Design Decisions

Some design decisions that have been made during devising the WMM deserve discussion and explanation:

Exactly One Synset For Each Memory Element
The design decision that each memory element should correspond to exactly one WordNet synset was made for simplicity:

The words the model will operate on are the German nouns listed in appendix A.1 which were used for conducting the experiment. However, the WordNet database is a model of the English language. So a good mapping needs to be found to map the German nouns to English synsets. A straightforward approach of translating each German noun into a English noun and then looking for synsets containing this English noun has one major disadvantage:

For example, consider the German noun *Tisch*. It can be translated into the English noun *table*. This noun appears in multiple synset, two of which are { *table, "a piece of furniture"*} and { *table, "a set of data arranged in rows and columns"*}. However, the second synset is not a meaning of the German word *Tisch* – if you translate it back, you get the German word *Tabelle*. So stimulating the synset { *table, "a set of data arranged in rows and columns"*} for the German noun *Tisch* is clearly inaccurate.

Due to this problem, the approach of mapping each German noun to exactly one corresponding WordNet synset was chosen. By doing so, the problem mentioned above does not occur. However, the ambiguity of several words cannot be taken into account. For example, when seeing the word *mouse*, one might either think of the animal or of the computer device. By determining only one corresponding synset, effects arising due to this ambiguity cannot be modeled.

On the other hand, using exactly one synset for each memory element simplifies the encoding and retrieval processes: If there were several synsets associated to one memory element, one would have to make a decision regarding the starting values for spreading. Will each synset receive all of the base level activation of the memory element or just a fraction? If it receives only a fraction, will the base level activation be equally distributed among the synsets or will there be a weighted distribution? And what if there are some memory elements with only one associated synset and some memory elements with maybe five associated synsets? The memory element having more associated synsets might receive more spreading activation than the memory element having only one associated synset. Are their activation values then still comparable? Moreover, are different memory elements allowed to share some common synsets? By using exactly one synset per memory element, these questions and design decision are avoided and the model is kept simple.

Time Measurement in Minutes
In the WMM, time is measured in minutes since the experiment start. This time unit has been chosen because it best reflects the time difference between encoding and retrieval of a word, which is usually between 25 seconds and 112 seconds. By using seconds as time unit, the range of values would have been too large, whereas by using minutes it is between and 0.42 and 1.87. As it can be seen in figure 2.1 in section 2.1.1, for a time difference $t_c - t_k$ of one, the base level activation is zero. So by choosing seconds as time unit, the base level activation of literally all memory elements would have been negative, whereas by choosing minutes as time unit, there are both elements with positive and elements with negative base level activation.

Initial Encoding Of Memory Elements
All memory elements are initially encoded at $t = -60$ min. If no initial encoding is provided, the activation of previously unlearned memory elements will always equal negative infinity. However, this is highly implausible, since an activation of negative infinity implies that this word has never been seen before. But the words are all used in common language, so they have definitely been seen before. Since an activation value of negative infinity is implausible, the activation of each memory element should be initialized in some way. The most straightforward way of implementing this is adding an initial encoding somewhere in the past.

Simplification of Time Steps
All encodings during a learning phase and all retrievals during a retrieval phase are assumed to take place at the same point in time, respectively, and because of that, time steps only do happen between phases but not within them. This simplification is made to avoid an oversensitivity regarding time. As the words were placed randomly within a phase when conducting the experiment, their mean position within a phase is in this phase's center point. Since the reinforcement and the gap condition are only based on time steps between phases but not on time steps within phases, this simplification should not impair the model's predictions.

Computation of Spreading Activation
In the WMM, the starting value for the spreading process is determined dependent on the base level activation. In LTM^C, for example, at the beginning of spreading, nodes which are currently in the context are activated. So when a person is asked which direction relation holds between Paris and London, the three nodes "direction relation", "Paris" and "London" will receive an initial activation which then is used for spreading. However, since the WMM tries to model effects arising in a recognition test, this approach seems to be rather impractical.

The only activation available in the WMM for use as spreading starting activation is the base level activation. The underlying assumption of using the base level activation here is that memory elements that are currently very "active" in memory influence their neighboring elements more than "silent" memory elements. However, the base level activation can be negative, and a negative base level activation does not imply that this memory element cannot be retrieved from memory. On the other side, spreading negative activation seems to be highly implausible because then received spreading activation might decrease the overall activation of a memory element. One possible effect could then be forgetting "boy" because "girl" has a negative activation value. In order to avoid effects like that, only positive values can be spread, and to allow even memory elements with a slightly negative base level activation to spread some activation to their neighbors, the spreading activation is lifted by a specific potential P_{spread}. The factor 10 by which this sum is multiplied accounts for the large number of neighbors a memory element from the WordNet database usually has: Since spreading activation is split up at each node, only little activation will reach nodes being two or three hops away. If spreading only adds values in the order of magnitude of 10^{-3} to the base level activation, there will be no noticeable effects of spreading, and hence the factor 10 was introduced to increase the effect of spreading.

No Noise Activation
In contrast to ACT-R and LTM^C, no noise activation is used. Since this model's purpose is only to predict mean values and a noise value usually would have an expected value of zero, this design decision will not effect the predicted mean values. However, by eliminating the influence of chance on the activation, a source of nondeterminism within the model is removed which in turn is an advantage for evaluation.

Modeling Workload Modes By Using Different Parameter Sets
Modeling different workload modes by simply using different parameter sets makes the least assumptions. Of course, it would be desirable to have only one single continuous workload parameter $w \in [0, 1]$ for distinguishing the different workload modes. However, since there is no a priori knowledge about which free parameters are influenced in which way by the different workload modes, modeling would become very complicated: For each equation, one would need to find a way in which w influences this equation without concrete knowledge of the implications. Moreover, the equations are dependent on each other and the interactions are not trivially to understand. By modeling the different workload modes with different parameter sets, the option to introduce such a parameter w is still left open: If the different parameter sets are compared, maybe some relations between the values can be observed.

A different approach would have been the use of linear regression. This probably would have yielded good results since all dimensions considered in the experiment consist of only two values (except the workload dimension which consists of three values). However, linear regression would estimate the dependent variables as linear combination of the dimensions. This means that the results obtained from linear regression will be less generalizable – for example, when introducing two more values for the reinforcement dimension, the predictions made by the linear regression model will probably be rather poor.

Interaction Of The Parameter Sets
The interaction of the different parameter sets was introduced because of the following reasoning:
Workload is assigned during encoding and not during retrieval. So when calculating the base level activation of a memory element, the workload mode which was active during encoding should be used. Although this was not the case in the experiment described in chapter 3, it could be possible that a memory element is encoded in different workload modes. To preserve this option, for each time of reinforcement t_k, the correct corresponding workload parameter set is used.

4.2 Implementation

The WMM was implemented in Java[1], using the MIT Java Wordnet Interface (v2.2.3, see http://projects.csail.mit.edu/jwi/), args4j (v2.0.21, see http://args4j.kohsuke.org/), Java CSV (v2.1, see http://www.csvreader.com/java_csv.php) and the "Apache Commons Math" library (v3.0, see http://commons.apache.org/math/). WordNet 3.0 (see http://wordnet.princeton.edu/) was used as database. See appendix B.2 for instructions on how to use the WMM implementation.

Figure 4.2 gives an overview of the most important classes of the WMM implementation and how they are associated. The *Starter* class contains the main() method and is therefore the program entry point. It processes the command line arguments and decides whether only a simulation shall be executed or whether a optimization should be started. In the former case, it transfers control to the *Tester* class which simulates one complete experiment setting with 24 participants. In the latter case, control is transferred to the *Optimizer* class which contains an algorithm for optimizing the model's parameters in order to predict the experiment's results. This algorithm will be described in section 5.1. The fourth important class inside the *framework* package is the *Parser* class which is responsible for parsing input files and for generating output files. Due to this function, both *Tester* and *Optimizer* use its services. All of this four classes implement the singleton design pattern since there should be only one object of them each.

As the *Tester* simulates a whole experiment setting with 24 participants, it needs to simulate 24 experiment runs. For generating new experiment runs, the *ExperimentRunGenerator* is executed. This python script has also been used to generate experiment runs for the actual experiment described in chapter 3. See appendix B.1 for more information on the *ExperimentRunGenerator*. Note that due to the usage of the experiment run generator script, which uses randomization, the WMM implementation also becomes nondeterministic, although the influence of randomization should be rather small in practice.

[1] using the Eclipse Indigo IDE

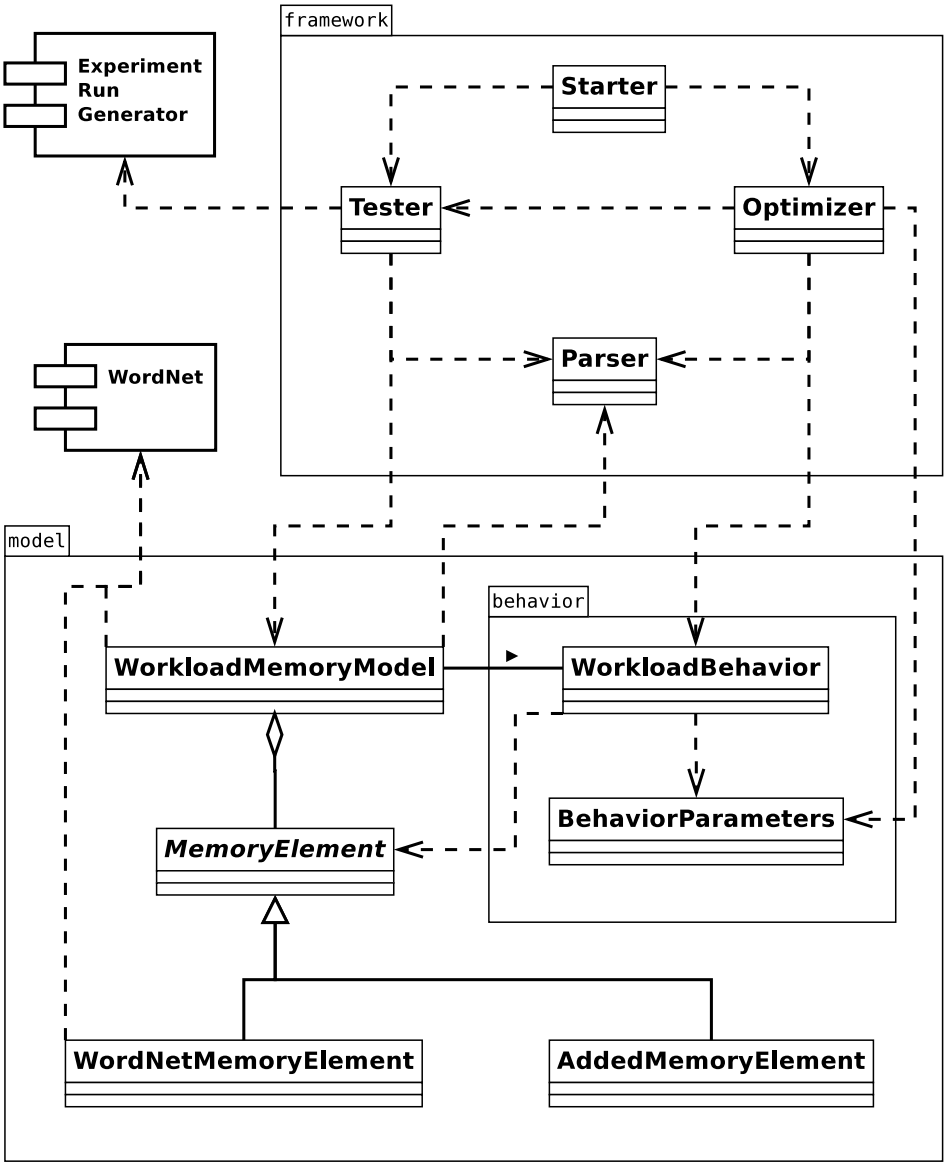

Figure 4.2: UML class diagram containing the most important classes of the WMM implementation. Dashed lines represent ”uses” relations.

Everything described until now is not part of the actual memory model but rather an automated way of using and accessing it. The actual memory model is contained in the *model* package. The *WorkloadMemoryModel* class (which is again implemented as a singleton) provides access to the model. It manages a set of memory elements and has an associated workload behavior. The *WorkloadBehavior* class implements this workload behavior and contains the equations and algorithms described in section 4.1.1. In total, there are three objects of the *WorkloadBehavior* class (one for each workload mode) which differ only by their parameter values. To ensure that there are exactly three *WorkloadBehavior* objects, a singleton-like access is used with three static *WorkloadBehavior* class attributes. The *BehaviorParameters* class encapsulates all free model parameters and is used for changing the parameters of a *WorkloadBehavior* when testing different parameter configurations during optimization.

The abstract *MemoryElement* class provides a partial implemented interface for the memory elements contained in the memory model. It is implemented by the two classes *WordNetMemoryElement* and *AddedMemoryElement*. The *WordNetMemoryElement* class implements a memory element that is connected to exactly one synset from the WordNet database. However, it became apparent that not all words used in the experiment could be mapped to an existing synset from the WordNet database. So an approach was needed to add some new elements. Since adding elements to the underlying WordNet database is a rather complex issue, and only a small number of words was involved, a different approach was chosen by implementing the *AddedMemoryElement* class. It behaves exactly the same as the *WordNetMemoryElement* class, but internally, the parent/child pointers are not retrieved by looking for hypernyms/hyponyms in the WordNet database, but by loading them from a configuration file on program start.

As it became apparent, adding new elements was necessary for another reason: The cluster words used in the experiment were chosen from Battig & Montague (1969) [18] who listed the most popular responses people make when asked for a word in a specific category (e.g. "a carpenter's tool"). The responses retrieved in their paper, however, are not based on any hierarchical, linguistical word database like WordNet. As a result, the distances in the WordNet database of the cluster words used in the experiment (e.g. pliers, hammer, saw, chisel, plane and nail for the "a carpenter's tool" category) were rather large. Figure 4.3 illustrates this problem: Not only are the cluster words far apart from each other, but also the number of edges for each intermediate node is relatively large. This also means that nearly no spreading activation would reach another cluster word via the hypernymy/hyponymy hierarchy for some clusters due to the large distances and the large degree of intermediate nodes.

To solve this problem, a new memory element was added ("popular tools") and the cluster words plus four more words with a high ranking in the category from Battig & Montague (1969) [18] were added as child nodes. The result is illustrated in figure 4.4. By modifying the database in this way, more direct paths between the cluster words are established in addition to the already existing connections. This was done for 20 of the 24 clusters used in the experiment. Of course, this is not an elegant solution, but it was necessary to modify the underlying database in this way in order to ensure that the spreading actually can work. However, for future research

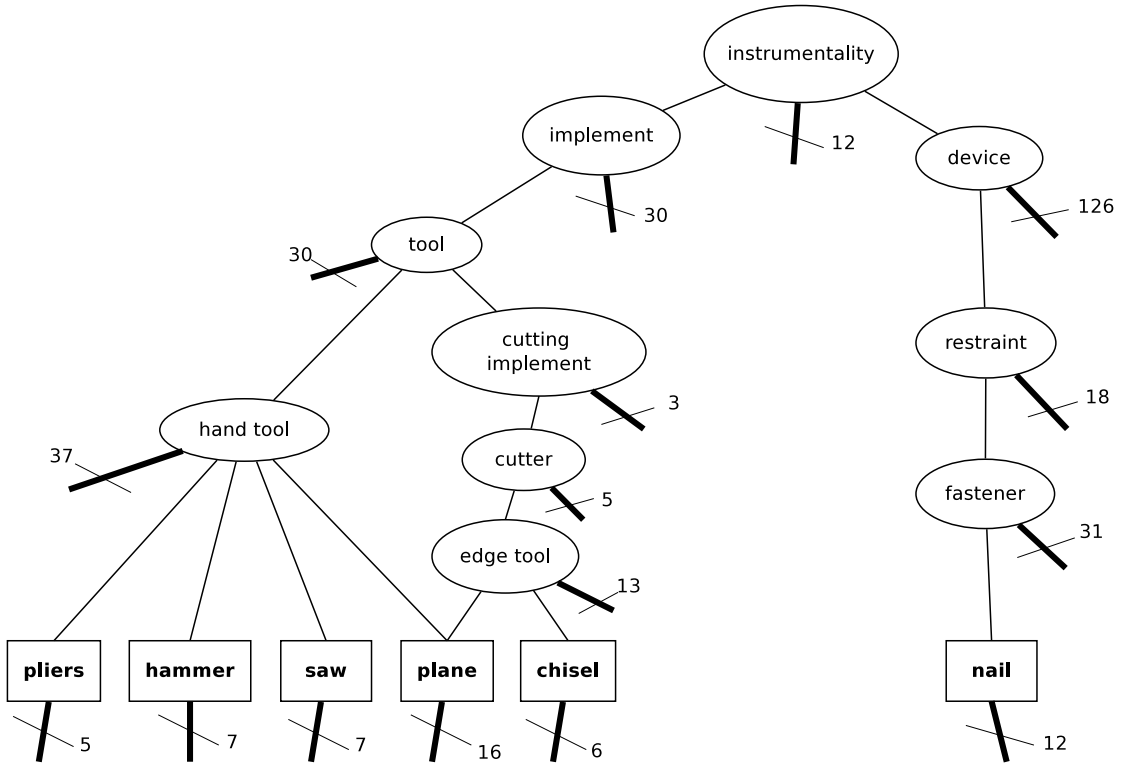

Figure 4.3: Structure of the WordNet noun hierarchy regarding the six cluster words from the ”a carpenter’s tool” category. Thick lines represent other hyponym relations of a synset, the annotated number represents the quantity of these relations.

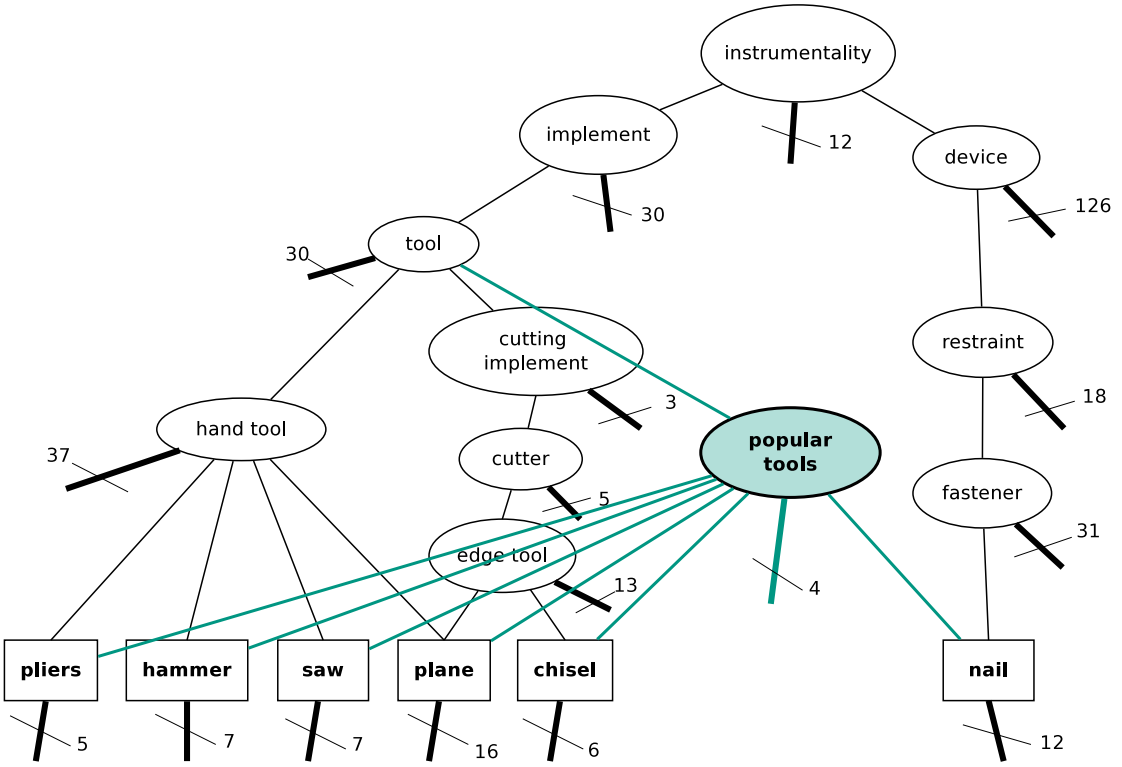

Figure 4.4: This figure illustrates the solution approach for the problem regarding cluster words in WordNet.

it would be advisable to choose a database different from WordNet. One possible database could be ConceptNet (see http://conceptnet5.media.mit.edu/) because it is more oriented on everyday knowledge than on linguistical word hierarchies.

To round out the description of the WMM implementation, an example program execution is described: The program is started with the intend to optimize the parameters. The *Starter* class parses the command line arguments and hands them on to the *Optimizer* singleton object. The *Optimizer* loads information like the ranges of valid parameter values by using the *Parser* singleton object. After three sets of parameters (encapsulated as *BehaviorParameter* objects) have been generated (one for each workload mode), they are passed on to the *WorkloadBehavior* objects. When the *WorkloadBehavior* objects have adopted their new parameters, the *Optimizer* invokes a simulation method on the *Tester* singleton object. The *Tester* calls the *ExperimentRunGenerator* to create new input data which is then read by again using the *Parser*. The *Tester* feeds the *WorkloadMemoryModel* with the first word to encode. Since this is the first time the *WorkloadMemoryModel* singleton object is used, it initializes by loading all memory elements appearing during the experiment into memory and reinforcing them at $t = -60$ minutes. Moreover, the added memory elements are parsed and added to the memory as well as the added connections between memory elements. These steps are again supported by the *Parser*. After each time step, the *WorkloadMemoryModel* calculates the current spreading activation by referring to the currently active *WorkloadBehavior*. Moreover, on each retrieval, the *WorkloadBehavior* is used for computing the retrieval probability and the expected response time. This is done by referring to the *MemoryElement* which

is holding both the times of reinforcement for calculating the base level activation, and the current spreading activation. The *Tester* receives all this retrieval probabilities and expected response times and hands them on to the *Optimizer* which can then judge whether the chosen parameter configuration is considered to be good or bad. Finally, when the optimizing algorithm is done, the results are prompted on the screen and written to a file by using the *Parser*.

4.3 Summary

In this chapter, a memory model was devised for predicting the effects observed during the experiment. The WorkloadMemoryModel is based on the ACT-R theory, uses a spreading approach similar to the one implemented in LTM^C and operates on the WordNet database. However, the WordNet database has turned out to be not an optimal choice for this utilization since words that are closely related in everyday knowledge (e.g. hammer and nail) are far apart from each other in the WordNet noun hierarchy. For further research, other databases like ConceptNet should be considered for usage. However, for the scope of this thesis, little additions were made to the WordNet database in order to use it properly. The following chapter 5 describes how this model is evaluated.

5. Evaluation

After devising the WMM, it is important to evaluate it by comparing the predicted values with the values measured in the experiment. This is necessary for answering the question whether the devised model can predict the observed effects and hence can be called a good model of human memory performance.

5.1 Optimization Algorithm

In order to optimize the model's parameters, a genetic algorithm was used. Genetic algorithms mimic the natural evolution process. They maintain a population of possible solutions, and for each of these possible solutions, a fitness value is computed. Now in every step, a new population element is produced – either by mutating a given one ("asexual reproduction") or by mating two given ones ("sexual reproduction"). The new offspring is added to the population and its fitness value is computed. In each step, one or two elements of the population are selected for reproduction and one element of the population is selected for removal. A high fitness value corresponds to a high probability of reproduction and a low probability of removal, whereas a low fitness value corresponds to a low probability of reproduction and a high probability of removal ("survival of the fittest"). After a fixed number of steps or when a specific fitness quality is reached, the algorithm terminates.

The optimization algorithm used in this thesis maintains one population of possible parameter configurations for each of the three workload modes. The size of these populations can be configured by the user, its default value is 100. The three populations are assumed to be independent of each other.

Each parameter configuration consists of values for decay d, intercept time I, retrieval threshold τ, spreading potential P_{spread}, distractor factor f and retrieval sensitivity s.

The fitness value is obtained by performing a simulation with this parameter configuration and by comparing the simulation results with the results of the experiment. Only mean values are compared, the standard deviation is ignored. The fitness value is then computed by adding up the relative errors for the three dependent variables

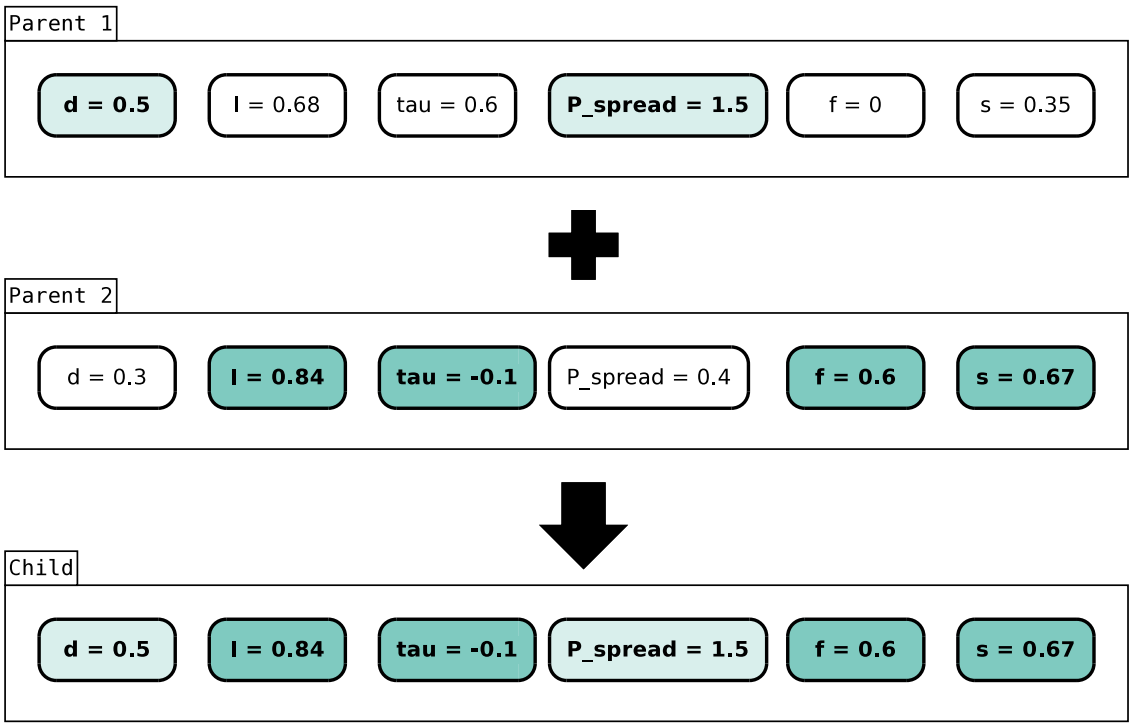

Figure 5.1: Mating operation of the optimization algorithm: The parameter values from parent 1 and parent 2 are combined to obtain a new child configuration.

response time, hit rate and false alarm rate. Weighting of these categories can be adjusted by the user. Per default, each of the listed variables has a weight of 1. The above-named sum is then negated since a low sum of relative errors should result in a relatively high fitness value and vice versa.

Mating of two parameter configurations is implemented by selecting a random number of parameters from the first parent and combining them with the missing parameters taken from the second parent. This results in a new parameter configuration that is added to the population. See figure 5.1 for an illustration.

Mutation of a parameter configuration is implemented by selecting a random number of parameters from the parent and varying the values of the selected parameters by adding a normal-distributed random variable. The resulting parameter configuration is then added to the population. See figure 5.2 for an illustration.

The genetic algorithm used for optimizing the WMM terminates after a fixed number of iterations (per default after 1000 iterations).

For more information on genetic algorithms, see Mehlhorn & Sanders (2008) [28] and Russell & Norvig (2010) [29].

Note that the computation of the fitness value is not completely deterministic, although the WorkloadMemoryModel itself is: The WMM implementation uses the ExperimentRunGenerator script to generate new experiment runs during a simulation. This script uses randomization to assign the words to the different dimensions (e.g. gap). In general, this randomization should not have a major influence on the resulting fitness values since for computing the fitness values, 24 experiment runs are simulated. Nevertheless, after termination of the optimization algorithm, it is

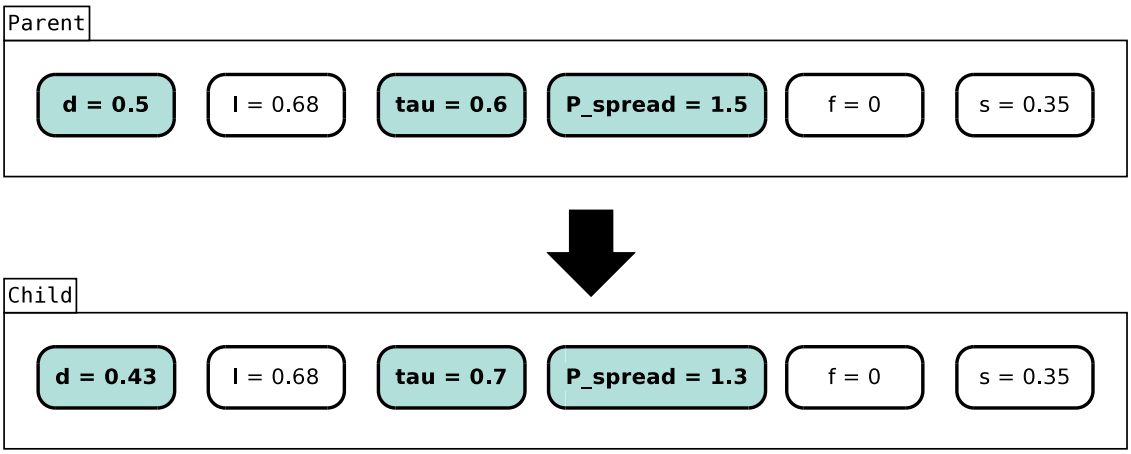

Figure 5.2: Mutating operation of the optimization algorithm: A random number of parameter values from parent is selected and randomly changed to obtain a new child configuration.

advisable to consider not only the parameter configuration with the highest fitness value, but also parameter configurations with comparable fitness values.

5.2 Optimization Results

Due to the relatively large number of free parameters, the model will not be optimized in one huge optimization run, but rather by using a step-by-step approach. This ensures the interpretability of the resulting parameter configurations since only little information is added during each evaluation step.

First, response time and hit rate in the dimensions reinforcement and gap will be evaluated, considering only singleton target words. This will be done with spreading deactivated. Second, the question will be analyzed whether the parameter configurations obtained in the first evaluation step are able to reproduce the differences in the workload dimension. Third, the cluster dimension will be evaluated by introducing the spreading mechanism and by considering the false alarm rate. Fourth, the target and cluster dimensions regarding response time will be evaluated by introducing a distractor factor. Finally, the improvements and impairments made in each step will be analyzed.

Each of the steps listed above aims at reproducing the empirical data gathered in the experiment. Two aspects are evaluated: the qualitative and the quantitative prediction of observed effects. Whereas the qualitative prediction of effects is evaluated by analyzing the differences of predicted values within one dimension, the quantitative prediction is evaluated by analyzing the fitness values obtained in the optimization, with a high fitness value indicating a good quantitative fit against the experiment data.

Parameter Configurations For Reinforcement-Gap-Optimization				
Workload Mode	d	τ	s	Fitness Value
None	0.500	-0.337	0.352	-0.068
Low	0.505	0.031	0.749	-0.081
High	0.798	0.218	0.965	-0.109

Table 5.1: Parameter configurations for the three workload modes as yielded by the optimization algorithm, and their corresponding fitness values. Note that the decay parameter d was set to 0.5 for workload mode None a priori (since this is a standard value that has emerged in the ACT-R community), and was therefore not optimized.

5.2.1 Evaluation of Reinforcement and Gap Dimension

As a first evaluation step, the model was optimized for reproducing the effects of reinforcement and gap which were observed in the experiment's results. Spreading was deactivated and only singleton target words were considered. The optimization algorithm was run for optimizing against the given response time and hit rate values, whereas false alarm rates were ignored for this evaluation step. Both response time and hit rate were assigned equal priority.

The intercept time I was set to 0.680 seconds which has emerged as reasonable value in several preliminary tests. Intercept time was kept constant since it is interpreted as time needed for perception and motor reaction and should therefore not be influenced by cognitive workload. Moreover, the distractor factor f was set to zero since it influences only the distractor response time and for this evaluation step, only target words were considered. Furthermore, the spreading potential P_{spread} was not optimized since spreading was disabled.

Hence, only the parameters d, τ and s were left for optimization. Table 5.1 shows their values as determined by the optimizing algorithm as well as the fitness values of these parameter sets. [1]

Tables 5.2 and 5.3 show the results obtained in the experiment and the predictions made by the model when configured with the parameter sets from table 5.1. Figure 5.3 and 5.4 illustrate these results using bar charts.

As the tables and figures indicate, the model can predict the effects observed in analyzing the experiment's results:
Response time of words retrieved with a gap is predicted to be higher and their hit rate is predicted to be lower than of words retrieved without a gap. Response time of words reinforced twice is predicted to be lower and their hit rate is predicted to be higher than of words reinforced once. Moreover, the model does not only qualitatively predict these results, but also quantitatively. This can be proved by considering the fitness values which are computed as the negated sum of relative errors:
As denoted in table 5.1, the three parameter sets have a fitness value of -0.068, -0.081 and -0.109, respectively, when evaluated against their corresponding workload

[1] The WMM was started with the command line arguments *-wFA 0 -maskNone 001001 -maskLow 101001 -maskHigh 101001 -filterRT 1133 -filterHit 133 -noSpreading.*

Response Times For Workload Mode None				
	reinf 1, gap 0	reinf 1, gap 1	reinf 2, gap 0	reinf 2, gap 1
Experiment Data	0.900	0.987	0.795	0.867
Optimized Model	0.860	0.926	0.799	0.850
Response Times For Workload Mode Low				
	reinf 1, gap 0	reinf 1, gap 1	reinf 2, gap 0	reinf 2, gap 1
Experiment Data	1.001	1.060	0.941	1.031
Optimized Model	0.986	1.070	0.903	0.973
Response Times For Workload Mode High				
	reinf 1, gap 0	reinf 1, gap 1	reinf 2, gap 0	reinf 2, gap 1
Experiment Data	1.136	1.187	1.114	1.131
Optimized Model	1.072	1.231	0.997	1.131

Table 5.2: Mean values for response time, both as measured in the experiment and as predicted by the model.

Hit Rates For Workload Mode None				
	reinf 1, gap 0	reinf 1, gap 1	reinf 2, gap 0	reinf 2, gap 1
Experiment Data	0.868	0.736	0.972	0.889
Optimized Model	0.867	0.643	0.958	0.888
Hit Rates For Workload Mode Low				
	reinf 1, gap 0	reinf 1, gap 1	reinf 2, gap 0	reinf 2, gap 1
Experiment Data	0.597	0.486	0.826	0.618
Optimized Model	0.597	0.446	0.728	0.617
Hit Rates For Workload Mode High				
	reinf 1, gap 0	reinf 1, gap 1	reinf 2, gap 0	reinf 2, gap 1
Experiment Data	0.563	0.389	0.729	0.451
Optimized Model	0.562	0.377	0.648	0.493

Table 5.3: Mean values for hit rate, both as measured in the experiment and as predicted by the model.

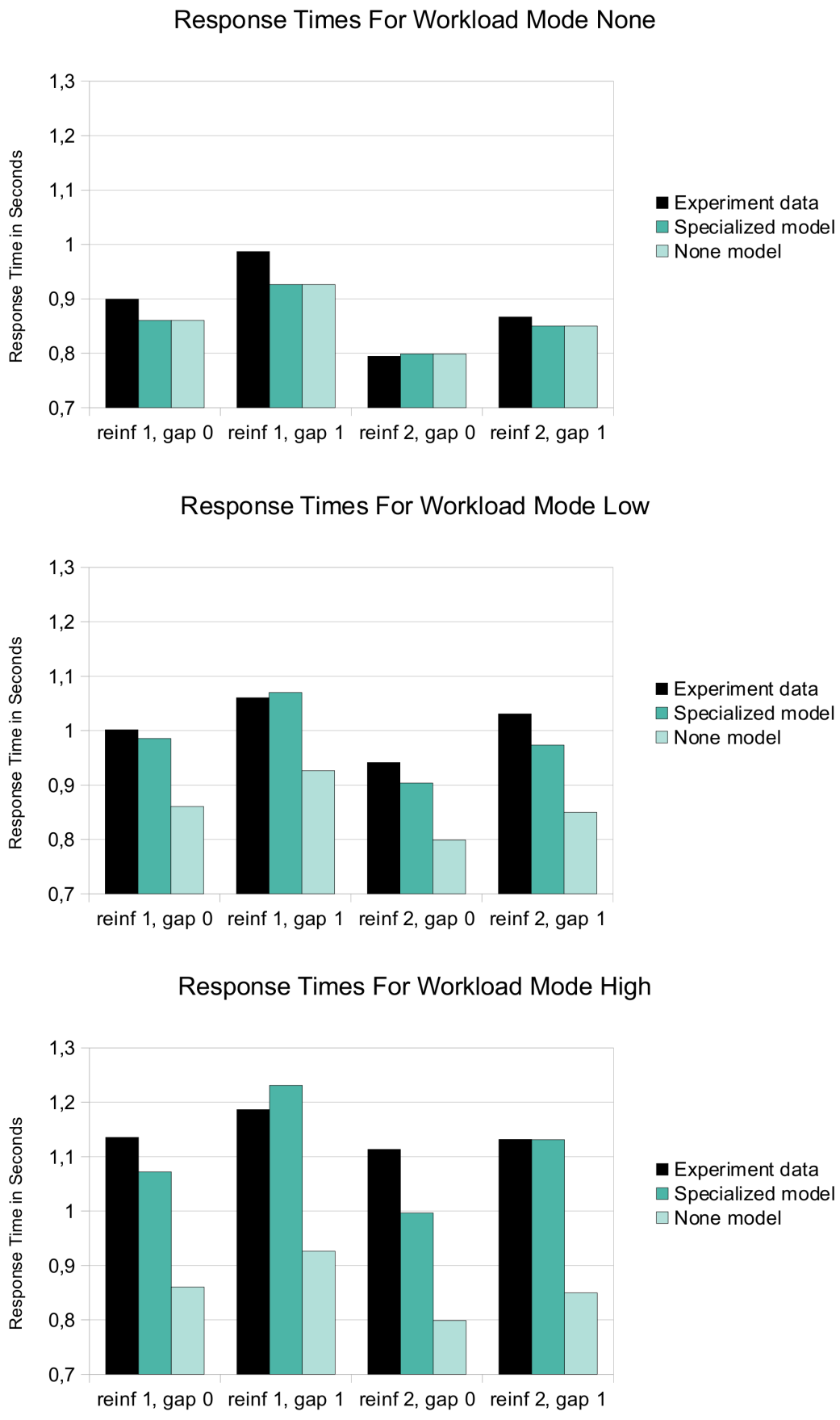

Figure 5.3: This graphs illustrates the results listed in table 5.2.

mode. As explained above, the fitness value is a negated weighted sum of mean relative errors made for the dependent variables response time, hit rate and false alarm rate. In this case, the fitness value is the negated sum of the mean relative error for response time and the mean relative error for hit rate, with both having a weight of 1. This means that the average relative error the model makes is around three to six percent which can be considered as a good fit against the experiment data.
However, as it can be seen seen for example in figure 5.3, the predictions made by the WMM are good but not perfect, and there is definitely potential for improvement.

In order to evaluate the approach of using three independent parameter sets, the parameter set for workload mode NONE was used as baseline configuration. Its fitness value when applied to the workload modes LOW and HIGH was computed as -0.491 for workload mode LOW and as -0.868 for workload mode HIGH, respectively. This huge difference between the fitness value of the specialized parameter sets and the fitness value of the baseline configuration can also be seen in figures 5.3 and 5.4 where the predictions of the NONE model are plotted against the predictions of the specialized model and the actual experiment results.

The relatively small average relative error of the specialized parameter sets in contrast to the comparatively large relative error of the NONE parameter set when transferred to the workload modes LOW and HIGH indicates that the design decision of modeling different workload modes as independent parameter configurations was a reasonable decision.

By taking a closer look to the parameters, the following observations can be made:

- The decay parameter d increases from 0.500 for NONE, over 0.505 for LOW up to 0.798 for HIGH. This seems plausible, because a higher value of d results in faster forgetting. Since the hit rate declines under higher workload, faster forgetting under higher workload seems to be a reasonable explanation approach.

- The retrieval threshold τ also increases: from -0.337 for NONE, over 0.031 for LOW to 0.218 for HIGH. This also seems reasonable: τ gives a threshold value for the activation of a memory element. Retrieval probability of a memory element at threshold activation equals 0.5, so a higher threshold τ means that elements need a higher activation for being retrieved. This appears to be another plausible explanation approach for the decreasing performance under higher workload.

- The retrieval sensitivity s does also increase: from 0.352 for NONE, over 0.749 for LOW to 0.965 for HIGH. This is also in line with the previous findings: Since s influences the sensitivity of the retrieval probability function, a value of s that is close to zero will cause the retrieval probability to form a smooth transition between low and high probabilities, whereas a value of s that is close to one will result in a rather sharp transition. When interpreted this way, the parameter values listed above indicate that for higher workload the decision whether a word can be retrieved or not becomes more like a binary decision. The value of 0.352 for workload mode NONE seems to be plausible since values around 0.4 have emerged as reasonable values for this parameter in the ACT-R community.

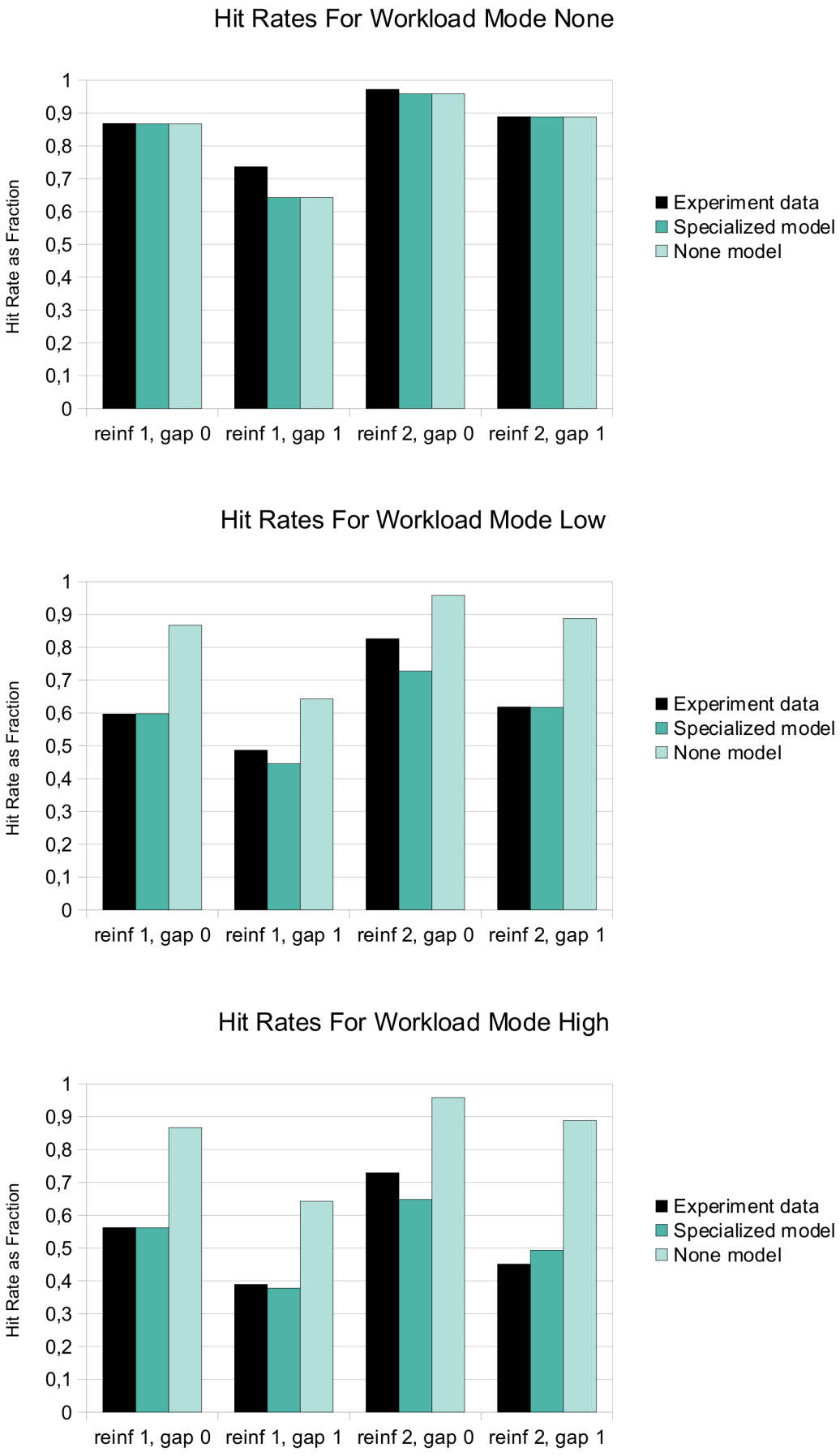

Figure 5.4: This graphs illustrates the results listed in table 5.3.

Response Times Across Workload Modes			
	None	Low	High
Experiment Data	0.887	1.008	1.142
Optimized Models	0.859	0.983	1.108
Hit Rates Across Workload Modes			
	None	Low	High
Experiment Data	0.866	0.632	0.533
Optimized Models	0.839	0.597	0.520

Table 5.4: Mean values of response time and hit rate across workload modes, both as measured in the experiment and as predicted by the model.

Fitness Values Across Workload Modes		
Workload Mode	Specialized Parameter Set	None Parameter Set
None	-0.063	-0.063
Low	-0.080	-0.476
High	-0.054	-0.822

Table 5.5: Fitness values for the parameter sets when predicting mean response time and mean hit rate for the listed workload modes.

5.2.2 Evaluation of Workload Dimension

Now that reinforcement and gap dimensions are evaluated, the next dimension to consider is the workload dimension. The model needs to be evaluated for its capability of reproducing the workload effects observed in the experiment results.

Again, only singleton target words are considered and spreading is deactivated. The parameters estimated in the previous section are used, and the resulting mean response time and mean hit rate for each of the workload modes is compared to the experiment data. Table 5.4 shows the predicted mean values and the ones measured in the experiment analysis. Figure 5.5 illustrates how they reproduce the experiment results in comparison to always using the None parameters. As it can be easily seen, the specialized parameter sets yield much better results.

The fitness values for each of the parameter configurations are denoted in table 5.5. Again, it can be seen that the fit for the specialized parameter sets is relatively good, whereas transferring the None model to workload modes Low and High results in relatively poor fitness values.

This step of evaluation showed that the model is also capable of modeling different workload modes.

5.2.3 Evaluation of Cluster Dimension

In the previous steps, two important aspects of the experiment and the model have been left out: The false alarm rate and the difference between singleton and cluster words.

This section aims at reproducing the differences between singleton and cluster words when considering only the false alarm rate. As a first attempt of reproducing the

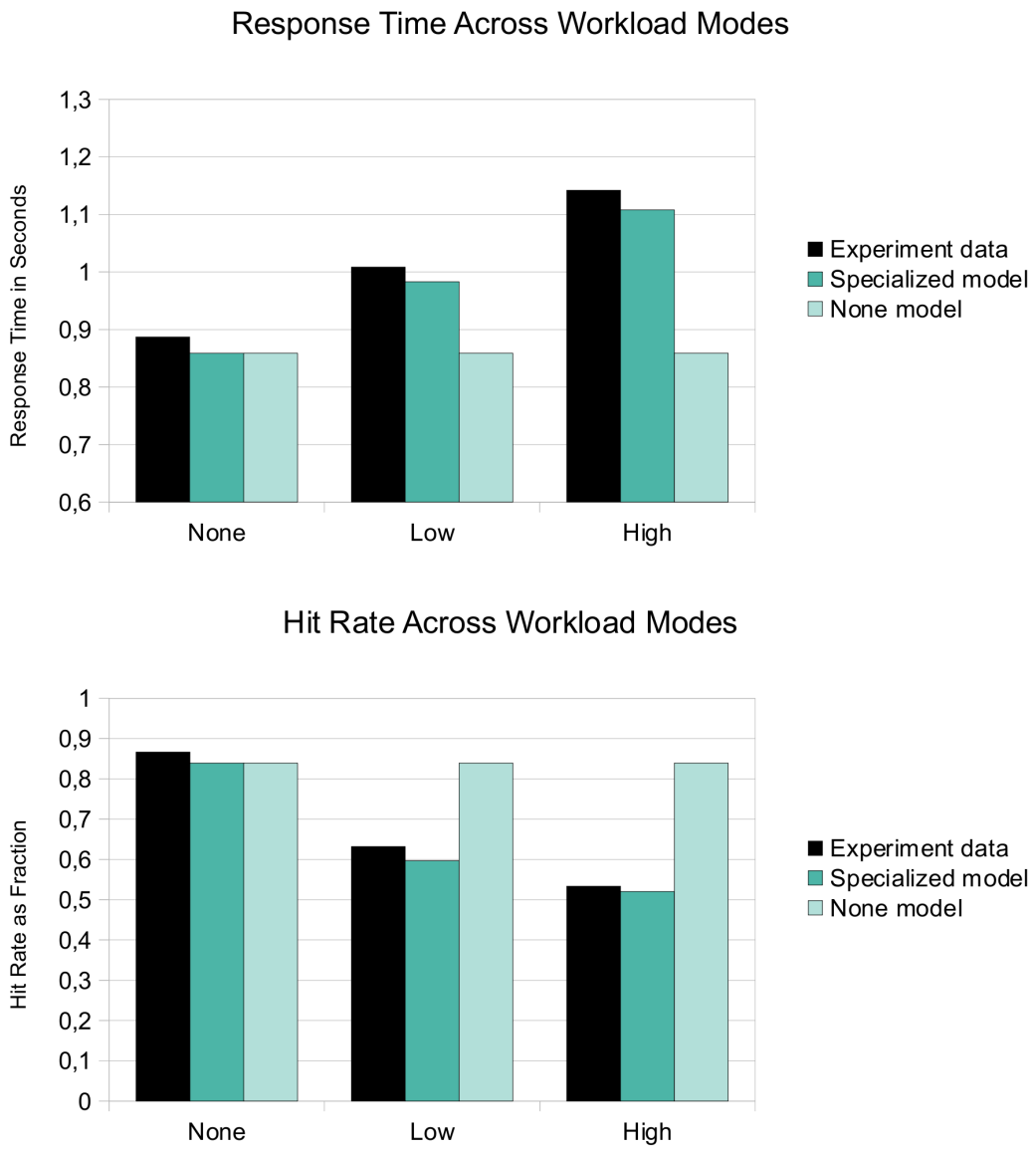

Figure 5.5: These bar charts illustrate the results listed in table 5.4.

Parameter Configurations of Cluster-Optimization					
Workload Mode	d	τ	s	$\boldsymbol{P_{spread}}$	**Fitness Value**
None	0.500	-0.337	0.352	**2.012**	**-0.219**
Low	0.505	0.031	0.749	**2.550**	**-0.171**
High	0.798	0.218	0.965	**2.983**	**-0.156**

Table 5.6: Parameter configurations for the three workload modes as yielded by the optimization algorithm when optimizing for singleton and cluster words, and their corresponding fitness values. Parameter values in grey cells were taken from the previous evaluation step, and were therefore kept fixed.

cluster effect on the false alarm rate, the parameter sets obtained in the previous sections were used and spreading was deactivated. As expected, the results were relatively poor with fitness values of -0.743 for None, -0.448 for Low and -0.360 for High.

As a next step, the model was optimized with spreading activated. This introduced a new parameter, the spreading potential P_{spread}, that had to be estimated. The parameters determined in previous evaluation steps were kept fixed and only P_{spread} was optimized.[2] The optimization results are denoted in table 5.6.

Table 5.7 compares the results yielded by both the baseline model and the spreading model to the actual experiment results. These comparison is illustrated in figure 5.6.

As it can be seen both in the table and in the bar charts, the model is capable of predicting the effect qualitatively when spreading is activated: The predicted amount of false alarms differs markedly when comparing singleton words and cluster words. However, the quantitative prediction appears to be not very accurate as indicated by the fitness values of -0.219 for None, -0.171 for Low and -0.156 for High. But at least, a marked improvement regarding the fitness values can be observed when comparing these fitness values to the ones obtained for the "no spreading" condition.

So what are the reasons of this rather imprecise quantitative prediction? As one would expect, there are several possible reasons:

- In the first evaluation step, the model has been optimized for predicting mean response time and mean hit rate. The parameters obtained in this first evaluation step might be a good fit for response time and hit rate, but not for false alarm rate. For example, response time and hit rate are in a different order of magnitude than false alarm rate (0.40 to 1.20 in contrast to 0.02 to 0.16). The fitness value is computed dependent on the relative error. Whereas an absolute error of 0.05 in hit rate leads to a comparably small relative error, the same absolute error in false alarm rate leads to a very large relative error. Because of this, it might be necessary to estimate all parameters together when optimizing for false alarm rate instead of using parameters optimized for hit rate.

[2] The WMM was started with the command line arguments *-wRT 0 -wHit 0 -maskNone 000100 -maskLow 000100 -maskHigh 000100 -size 50 -it 500.*

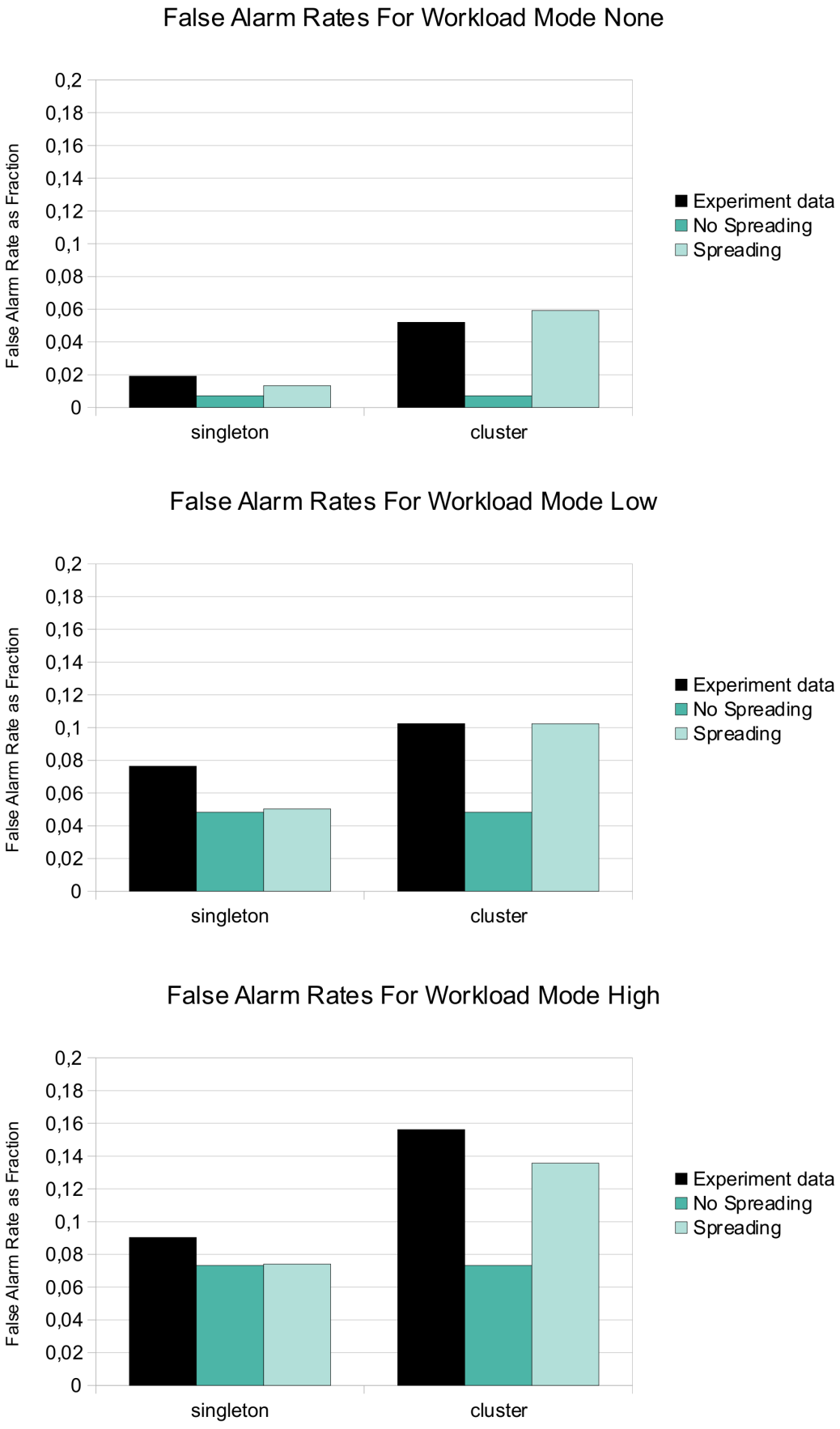

Figure 5.6: This graphs illustrates the results listed in table 5.7.

False Alarm Rates For Workload Mode None		
	singleton	cluster
Experiment Data	0.019	0.052
No Spreading	0.007	0.007
Spreading	**0.013**	**0.059**

False Alarm Rates For Workload Mode Low		
	singleton	cluster
Experiment Data	0.076	0.102
No Spreading	0.048	0.048
Spreading	**0.050**	**0.102**

False Alarm Rates For Workload Mode High		
	singleton	cluster
Experiment Data	0.090	0.156
No Spreading	0.073	0.073
Spreading	**0.074**	**0.136**

Table 5.7: Mean false alarm rates measured in the experiment and predicted by the model – either with spreading deactivated or with spreading activated.

- Another possible reason are the degrees of freedom of the implemented spreading algorithm: Maybe, the spreading threshold τ_{spread} or the factor 10 that was introduced for computing the starting value for the spreading algorithm need to be treated as free parameters that can also be optimized. This might lead to a better fit than keeping them fixed or dependent on another parameter.

- Of course, it is also be possible that the spreading approach in general is unsuitable for this area of application. This, however, is rather unlikely since the effect can be predicted qualitatively. Moreover, spreading is implemented in different cognitive architectures and memory models (e.g. ACT-R and LTM^C) and has proved to be a useful concept.

- As noted in the implementation section, the underlying WordNet database proved to be rather unsuited for this area of application due to the large distance of cluster words. Perhaps the predictions become more precise when a different database is used which is more suited for working with semantically clustered words.

When analyzing the parameter configurations given in table 5.6, an interesting tendency regarding the spreading potential P_{spread} can be observed: It rises from 2.012 for None, over 2.550 for Low to 2.983 for High. This tendency can be interpreted as spreading becoming more important under higher workload. This would be in line with the findings from the experiment in chapter 3 where we found that learning words in clusters seems to become more important with increasing workload. However, this tendency should be confirmed by further research.

Hit Rates For Workload Mode None			
	singleton	cluster	Fitness Value
Experiment Data	0.866	0.885	-
No Spreading	0.839	0.839	-0.042
Spreading	**0.838**	**0.869**	**-0.025**
Hit Rates For Workload Mode Low			
	singleton	cluster	Fitness Value
Experiment Data	0.632	0.726	-
No Spreading	0.597	0.597	-0.116
Spreading	**0.610**	**0.664**	**-0.060**
Hit Rates For Workload Mode High			
	singleton	cluster	Fitness Value
Experiment Data	0.533	0.618	-
No Spreading	0.520	0.520	-0.091
Spreading	**0.536**	**0.621**	**-0.005**

Table 5.8: This table shows the predictions for the hit rates regarding the cluster dimension as made by the model, and compares them to the experiment results.

Although the model was not optimized to reproduce the effects of cluster on hit rate, the predictions made for this dependent variable regarding the cluster dimension were also analyzed. The results of this analysis are listed in table 5.8 and illustrated in figure 5.7. Again, the observed effects are predicted qualitatively if spreading is activated. The predicted values, although slightly imprecise (as indicated by the bar charts), yield relatively good fitness values. This indicates that the WMM is generally capable of predicting cluster effects on hit rate.

5.2.4 Evaluation of Target Dimension

One parameter has still been excluded from all previous optimization and evaluation steps: the distractor factor f. This parameter influences the response time of words that have not been successfully retrieved from memory and therefore plays a major role in the difference between the response time of target and distractor words. In all previous evaluation steps, f has been set to zero.

As a next step, the optimization algorithm was run again with all previously determined parameters fixed and only f as free parameter. The differences between target and distractor words were considered both regarding singleton and cluster words.[3]

Table 5.9 shows the parameter configurations yielded by the optimization algorithm and their fitness values. As it can be seen, the fitness value of workload mode None is notably worse than the fitness values of workload modes Low and High. Table 5.10 denotes the predicted response times regarding cluster and target dimension. It is apparent that in workload mode None the predictions made when the distractor factor is optimized do not significantly differ from the predictions made by the

[3]The WMM was started with the command line arguments *-wFA 0 -wHit 0 -maskNone 000010 -maskLow 000010 -maskHigh 000010 -filterRT 3300 -size 25 -it 250.*

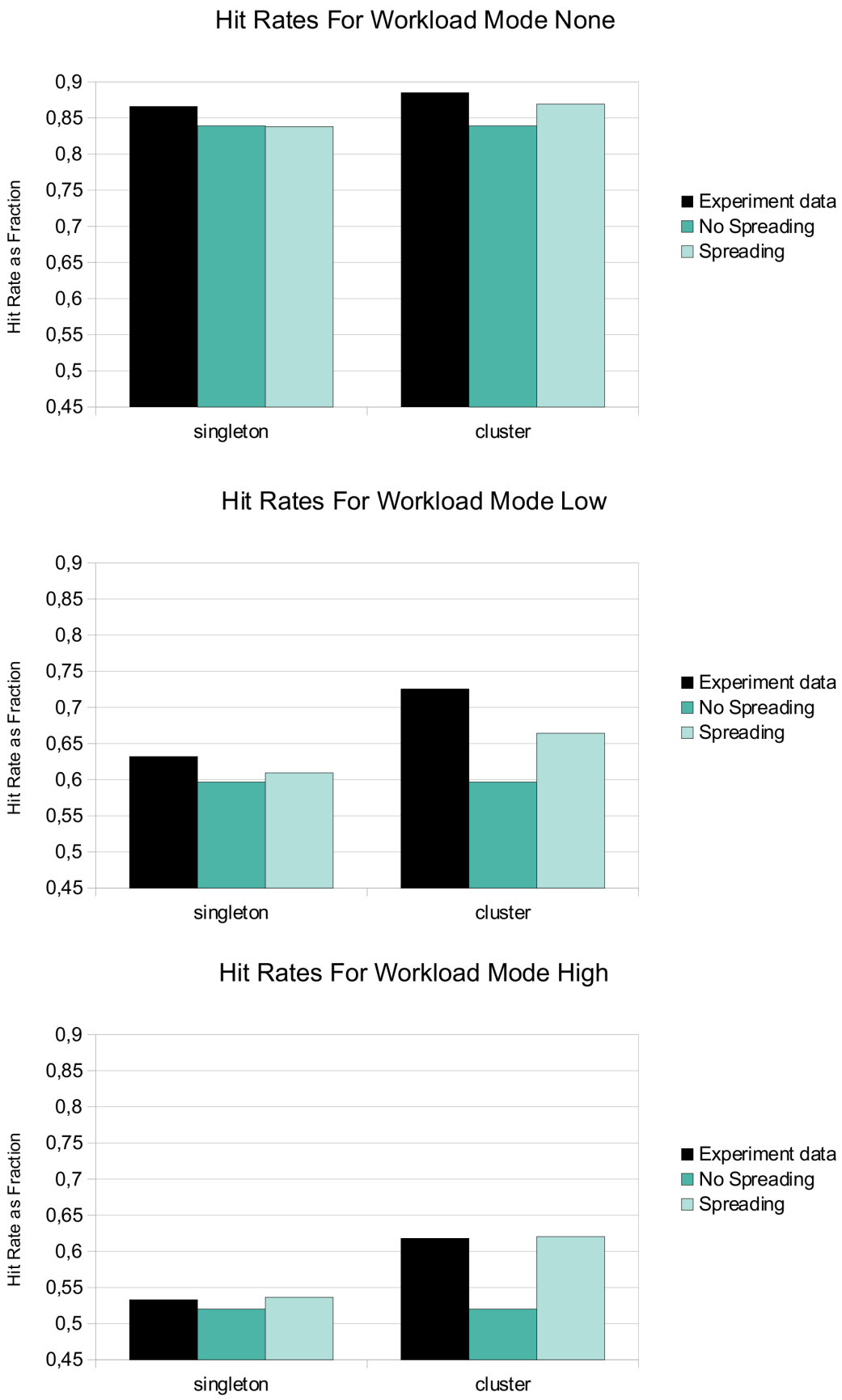

Figure 5.7: This figure illustrates the results from table 5.8.

Parameter Configurations of Target-Optimization						
Workload Mode	d	τ	s	P_{spread}	f	**Fitness Value**
None	0.500	-0.337	0.352	2.012	**0.004**	**-0.074**
Low	0.505	0.031	0.749	2.550	**0.219**	**-0.025**
High	0.798	0.218	0.965	2.983	**0.569**	**-0.034**

Table 5.9: Parameter configurations for the three workload modes as yielded by the optimization algorithm when optimizing for target and distractor words, and their corresponding fitness values. Parameter values in grey cells were taken from previous evaluation steps, and were therefore kept fixed.

parameter set from section 5.2.3. However, this comes as no real surprise since the distractor factor $f = 0.004$ obtained by the optimization algorithm is very close to zero, and therefore, the above-named parameter sets do not differ notably.

When analyzing the workload modes Low and High, a significant improvement by optimizing the distractor factor can be observed. This is also illustrated in figure 5.8: Predictions for distractor words become much better with optimized distractor factor. However, as it can be seen in table 5.10, the effect of cluster distractors having a higher response time than singleton distractors is not predicted distinctly – only a very slight tendency can be observed. This also holds for the general effect of target: In workload mode Low, the effect of distractor words having a higher response time than target words is correctly predicted. In workload mode None, this prediction is not distinct, and in workload mode High a converse tendency is predicted.

Introducing the distractor factor f improves predictions for workload modes Low and High, but the model is still not capable of predicting the observed effects distinctly. This again might be caused by the previous optimization for other dimensions and dependent variables. As the improvement in fitness values suggests, this approach of the activation value into the equation for distractor response time is a step in the right direction. However, this approach is clearly not satisfying, so further research is needed in trying to reproduce the observed effects.

When comparing the values of the distractor factor f, again a tendency can be observed: Its value rises from 0.004 in workload mode None, over 0.219 in workload mode Low to 0.569 in workload mode High. This would mean that the influence of activation on a distractor word's response time rises with increasing workload, which again can be interpreted as the response time difference between singleton and cluster distractors becoming larger with increasing workload. This tendency, however, was not observed in the experiment. Therefore, this parameter's development across the workload modes, although being consistent, is not very plausible.

Response Times For Workload Mode None				
	singleton		cluster	
	target	distractor	target	distractor
Experiment Data	0.887	0.913	0.891	1.018
No Spreading	0.859	0.870	0.859	0.870
Spreading	0.856	0.865	0.837	0.866
Distractor Factor	**0.856**	**0.863**	**0.841**	**0.863**

Response Times For Workload Mode Low				
	singleton		cluster	
	target	distractor	target	distractor
Experiment Data	1.008	1.017	0.976	1.067
No Spreading	0.983	1.186	0.983	1.186
Spreading	0.975	1.171	0.938	1.157
Distractor Factor	**0.981**	**1.018**	**0.946**	**1.021**

Response Times For Workload Mode High				
	singleton		cluster	
	target	distractor	target	distractor
Experiment Data	1.142	1.051	1.062	1.117
No Spreading	1.108	1.477	1.152	1.270
Spreading	1.093	1.473	1.134	1.263
Distractor Factor	**1.135**	**1.092**	**1.149**	**1.107**

Table 5.10: This table compares the response times for cluster and target dimension as predicted by the different model settings in contrast to the experiment data. "No Spreading" stands for the parameter configuration from section 5.2.1 with spreading deactivated, "Spreading" for the parameter configuration from section 5.2.3 section with spreading activated and "Distractor Factor" for the parameter configuration obtained in this section when optimizing the distractor factor, also with spreading activated.

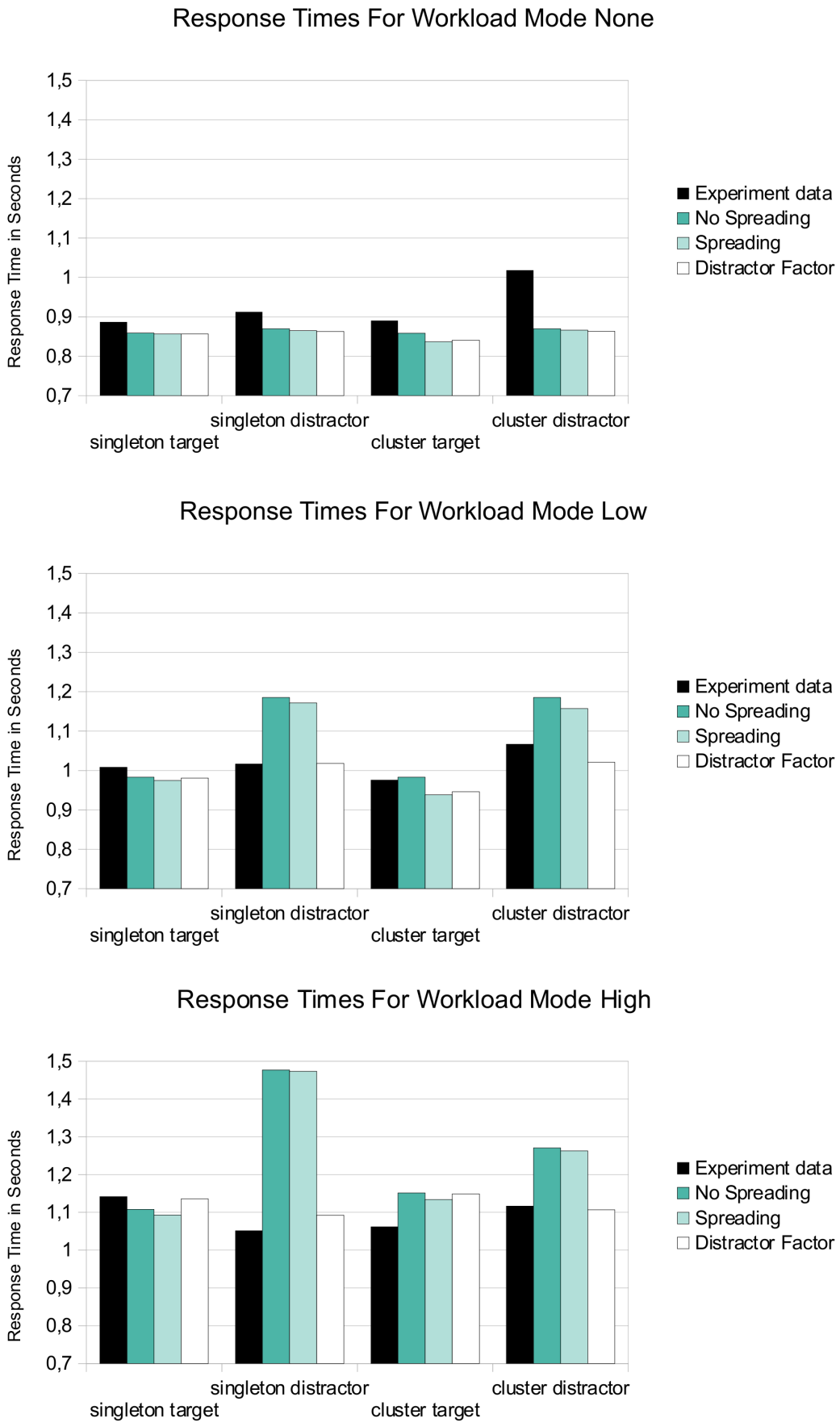

Figure 5.8: This figure illustrates the contents of table 5.10.

Response Time in Reinforcement and Gap Dimension			
	NONE	LOW	HIGH
Evaluation Step 1	-0.033	-0.030	-0.050
Evaluation Step 2	-0.034	-0.034	-0.060
Evaluation Step 3	-0.034	**-0.028**	**-0.015**
Hit Rate in Reinforcement and Gap Dimension			
	NONE	LOW	HIGH
Evaluation Step 1	-0.036	-0.051	-0.059
Evaluation Step 2	-0.037	-0.054	-0.069
Evaluation Step 3	-0.037	**-0.048**	-0.069
False Alarm Rate in Cluster Dimension			
	NONE	LOW	HIGH
Evaluation Step 1	-0.743	-0.448	-0.360
Evaluation Step 2	-0.219	-0.171	-0.156
Evaluation Step 3	**-0.289**	**-0.237**	**-0.111**
Response Time in Target and Cluster Dimension			
	NONE	LOW	HIGH
Evaluation Step 1	-0.065	-0.078	-0.164
Evaluation Step 2	-0.074	-0.077	-0.161
Evaluation Step 3	-0.074	-0.025	-0.034

Table 5.11: Fitness values as they have emerged through the evaluation steps. "Evaluation Step 1" corresponds to the parameter set obtained in section 5.2.1, "Evaluation Step 2" to the one obtained in section 5.2.3 and "Evaluation Step 3" to the one obtained in section 5.2.4.

5.2.5 Evaluation of Optimization Steps

In the previous sections, the parameters have been optimized step by step. Since parameters were added (P_{spread}) or changed (f) by proceeding this way, it is necessary to evaluate the impairment of previously optimized measurements. This is done in table 5.11.

When considering response time evaluated in the dimensions reinforcement and gap (for singleton target words only), a slight impairment can be observed after evaluation step 2 (optimizing spreading for false alarm rate regarding the cluster dimension). In workload modes LOW and HIGH, optimization of the distractor factor (evaluation step 3) leads to an improvement in fitness. Since the response times yielded by the model are always expected response times, a more accurate prediction of response time in case of a failure to retrieve improves also the expected values of the response time for target words.

When considering hit rate evaluated in the dimensions reinforcement and gap, again a slight impairment by introducing spreading can be observed. As probability of retrieval of target words is used to predict hit rates, and as probability of retrieval

is not dependent on the response time, the fitness values of evaluation step 2 and 3 should not differ since only response time is influenced in evaluation step 3. However, the fitness value in workload mode LOW decreases from −0.054 after evaluation step 2 to −0.048 after evaluation step 3. The only reasonable explanation of this effect is the nondeterminism of the underlying experiment run generator script which generates the data needed for performing simulation and hence for calculating fitness values. Although the influence of the randomization used in this script should be rather small, it seems to account for a difference of 0.006 in the fitness value.

This effect is even more obvious when considering false alarm rate in the cluster dimension: The difference between the fitness values after evaluation step 2 and evaluation step 3 are 0.070, 0.066 and 0.045, respectively. This can be explained by the way false alarm rate and fitness value are obtained: False alarm rate is computed as probability of retrieval of distractor words, whereas hit rate is computed as probability of retrieval of target words. This means that false alarm rate and hit rate are influenced in the same order of magnitude by the nondeterminism of the experiment run generator script. Fitness values are computed as weighted sum of average relative errors. Since the false alarm rate has markedly lower values than the hit rate, variations in similar order of magnitude will have larger impact on the relative error regarding the false alarm rate than on the relative error regarding the hit rate. In simplified terms, the fitness value of the false alarm rate is more sensitive to variations in the predicted false alarm rates because it operates on smaller numbers.

The values for response time evaluated in the dimensions target and cluster show that introducing the distractor factor f leads to a significant improvement in fitness for workload modes LOW and HIGH, as discussed in section 5.2.4. The fact that the previous introduction of spreading led to a slight improvement in workload modes LOW and HIGH is caused by the difference in activation between singleton and cluster words provided by the spreading activation.

When comparing the fitness values of the different workload modes, the following observations can be made: When considering hit rate, fitness values decrease with increasing workload, whereas for false alarm rate, the fitness values increase with increasing workload. For response time, no such tendency is observable. The absence of a general tendency (e.g. fitness values of NONE being always better than fitness values of LOW and HIGH) indicates, that the model is not specialized to one of the workload modes and only partially transferable to the remaining two.

5.3 Summary

By using a genetic optimization algorithm and a step-by-step approach, the model devised in chapter 4 has been optimized and evaluated. These are the key findings from this section:

- The effects of reinforcement and gap can be reproduced in all workload modes.
- Modeling different workload modes by using different parameter sets yields good results. These results are especially better than the results obtained by using one model (the one of workload mode NONE) for all workload modes.

- The effects of cluster can be reproduced qualitatively, but not qualitatively by using the implemented spreading algorithm. Results are markedly better with activated spreading in comparison to deactivated spreading.
- The effects of target cannot be predicted by the model accurately. The approach of considering activation (multiplied by a distractor factor f) when computing the response time generally leads to a better fit against the data obtained in the experiment. However, the results regarding the target dimension are clearly not satisfying.
- By using the step-by-step approach, the results of a previous step are only slightly impaired when doing the next evaluation step.

The main question stated in the introduction was: "Can models of human memory like the ACT-R declarative module be adapted to different workload settings by modifying the model's free parameters in a plausible way, or is it necessary to devise completely or partially new models for human memory under workload?" According to the evaluation, this question can be answered in the following way: It is in general possible to transfer a memory model to different workload settings. The parameter modifications necessary to adapt the model to these different workload settings are reasonable for all parameters except the distractor factor. However, since the effects of target could not be reproduced by using the distractor factor, this is not caused by the model's non-transferability to different workload modes, but rather by a general shortcoming regarding this point.

The second question stated in the introduction section was: "Moreover, if different workload modes can be simulated by modifying the model's parameters, how do these parameter modifications influence the model's behavior?" According to the evaluation, this question can be answered in the following way: The decay parameter d rises with increasing workload, which leads to a faster decline in base level activation and therefore to faster forgetting. Also the retrieval threshold τ rises with increasing workload. This means that memory elements need a higher activation value to be retrieved from memory. The retrieval sensitivity s, which controls the slope of the retrieval probability curve, also grows with increasing workload. This can be interpreted as decision making becoming more binary under high workload. Also the spreading potential P_{spread} rises with increasing workload, which suggests that spreading and therefore learning words as clusters become more important under high workload. Finally, the distractor factor f also has an ascending tendency with increasing workload. This can be interpreted as the response time difference between cluster distractors and singleton distractors becoming larger with increasing workload. However, this is not in line with the experiment results.

In total, the evaluation shows that the approach of transferring a memory model to different workload stage by modifying its parameters is reasonable, although the evaluated model still has potential for improvement.

6. Conclusion

In this thesis, the performance of human memory under cognitive workload was examined by conducting a psychological experiment. Moreover, a model for predicting the observed effects was devised.

The conducted experiment used the method of recognition test for measuring the memory performance and the method of divided attention to create three different workload modes which were labeled as NONE, LOW and HIGH. The secondary task used in the experiment was based on the switching task and had two levels of difficulty. As the experiment results show, there were significant effects in the dimensions workload, target, cluster, reinforcement and gap.

Based upon the ACT-R declarative module and LTM^C, the WorkloadMemoryModel (WMM) was devised which uses WordNet as underlying database. However, this database had to be modified to satisfy the model's needs.

As its evaluation showed, the WMM is capable of predicting the mean values of response time, hit rate and false alarm rate. As the differences between cluster words and singleton words can be predicted qualitatively, but not quantitatively, further research regarding the spreading approach is advised. The parameter values across the different workload modes follow an interpretable and plausible tendency. The single exception is caused by a general model weakness in predicting effects of target and can therefore not be assigned to the model's transferability to different workload modes.

In summary, this thesis proved that it is possible to model the influence of cognitive workload on human memory performance by using a model with modified parameters for each workload mode. However, the devised model is of course only a first step in the attempt to build workload-aware user interfaces: It operates on a rather artificial setting and the current workload is given a priori.

The results of this thesis leave space for further research:

- The effects observed in the cluster dimension are not fully explainable by theories like the Shared Time Model by Craik et al. (1996) [3]. More research

is necessary in order to include this effects in existing explanation theories or to come up with a new theory. Furthermore, some effects on response time require further investigation.

- Moreover, since the WordNet database has turned out to be unsuitable in this field of research, the model should be evaluated again when using a database more adapted to everyday knowledge (e.g. ConceptNet).

- The response time effect of target and the response time effect of cluster on distractor words cannot be predicted by the WMM. Further research is required to eliminate this shortcoming of the WMM.

- Furthermore, the WMM simply manages one parameter configuration for each workload mode. However, some tendencies could be observed regarding the parameters, e.g. the decay parameter d increases with higher workload. Further research using different secondary tasks could confirm or falsify these tendencies. In case they are confirmed, the attempt could be made to extract a workload factor w out of the WMM equations, so that there is only one parameter configuration for all workload modes and only the workload factor is changing across workload modes.

- Of course, also integration with other models should be a major aspect of further research. For example, the WMM can be combined with a workload recognizing system in contrast to giving the workload state a priori as it has been done for the scope of this thesis. Especially this integration with other models and systems is crucial for making a step towards the goal of workload-aware and adaptive user interfaces.

A. Experiment Information

A.1 Word Lists

A total number of six word lists was used for the experiment. Each word list contained a total number of 54 German nouns. 24 of them were cluster words that belonged to a cluster and 30 of them were singleton words that did not belong to a cluster. Three words of each cluster were used as target words and the remaining three words of that cluster were used as distractor words. Twelve singleton words were used as target words, twelve were used as distractor words and the remaining six singleton words were used as filler words that were learned but never retrieved.

A.1.1 Word List 1

- **Cluster 1**: Bier, Wein, Whiskey, Rum, Sekt, Wodka
- **Cluster 2**: Kuh, Löwe, Tiger, Hund, Katze, Pferd
- **Cluster 3**: Blau, Rot, Grün, Gelb, Schwarz, Weiß
- **Cluster 4**: Stuhl, Tisch, Bett, Sofa, Lampe, Schrank
- **Singletons**: Welt, Physik, Fleisch, System, Milch, Besitzer, Gesetz, Qual, Ecke, Eisenbahn, Schild, Hemd, Sprache, Material, Markt, Träne, Fell, Zelle, Monster, Spur, Bettler, Tinte, Lärm, Vorlesung, Kuss, Geruch, Flasche, Länge, Spiel, Verletzung

A.1.2 Word List 2

- **Cluster 1**: Eisen, Kupfer, Stahl, Zinn, Bronze, Nickel
- **Cluster 2**: Öl, Benzin, Kohle, Holz, Kerosin, Petroleum
- **Cluster 3**: Tante, Onkel, Vater, Mutter, Bruder, Schwester
- **Cluster 4**: Kirche, Tempel, Kapelle, Moschee, Schrein, Kloster

- **Singletons**: Verb, Katastrophe, Boss, Turm, Forschung, Strom, Balkon, Wolle, Lied, Mond, Wunder, Freund, Luft, Umfrage, Wärme, Kissen, Wand, Geheimnis, Hafen, Teddy, Vertrag, Geldbeutel, Atom, Armee, Jungfrau, Rauch, Form, Fenster, Flugzeug, Oase

A.1.3 Word List 3

- **Cluster 1**: Messer, Gabel, Löffel, Pfanne, Topf, Mixer
- **Cluster 2**: Arzt, Richter, Lehrer, Ingenieur, Schreiner, Händler
- **Cluster 3**: Jahr, Tag, Monat, Stunde, Minute, Woche
- **Cluster 4**: Brief, Novelle, Buch, Roman, Zeitung, Gedicht
- **Singletons**: Magnet, Stift, Schlüssel, Karotte, Mehrheit, Eigentum, Brei, Batterie, Prärie, Bildung, Kandidat, Lächeln, Fahrkarte, Maschine, Schock, Klavier, Spatz, Armut, Eimer, Getreide, Geschenk, Baron, Herd, Walzer, Sklave, Strafe, Patent, Winkel, Video, Stil

A.1.4 Word List 4

- **Cluster 1**: Arm, Bein, Kopf, Auge, Fuß, Nase
- **Cluster 2**: Apfel, Orange, Banane, Kirsche, Birne, Pfirsich
- **Cluster 3**: Berg, Hügel, Tal, Fluss, Felsen, See
- **Cluster 4**: Granate, Pistole, Gewehr, Bombe, Schwert, Knüppel
- **Singletons**: Verkleidung, Schaum, Hochzeit, Schule, Gerücht, Witz, Galerie, Geist, Stern, Mast, Weihnachten, Lagerfeuer, Keller, Heizung, Gefängnis, Proton, Wiege, Partei, Rassel, Rezept, Produkt, Seite, Absicht, Auto, Tabak, Reise, Baby, Fliege, Gewicht, Münze

A.1.5 Word List 5

- **Cluster 1**: Haus, Apartment, Hütte, Hotel, Villa, Schuppen
- **Cluster 2**: Diamant, Rubin, Smaragd, Saphir, Perle, Opal
- **Cluster 3**: Basketball, Tennis, Schwimmen, Fußball, Golf, Hockey
- **Cluster 4**: Salz, Pfeffer, Zucker, Knoblauch, Vanille, Zimt
- **Singletons**: Blut, Dorf, Licht, Schach, Fremder, Wissen, Koffer, Fußboden, März, Arie, Stille, Internet, Inhalt, Senf, Rabbi, Pfeil, Einwohner, Ersatz, Bild, Emotion, Richtung, Eiche, Muskel, Paar, Funke, Präsident, Schiff, Bürste, Jazz, Zeichen

A.1.6 Word List 6

- **Cluster 1**: Mord, Raub, Diebstahl, Überfall, Einbruch, Betrug
- **Cluster 2**: Hammer, Säge, Nagel, Hobel, Meißel, Zange
- **Cluster 3**: Hurrikan, Tornado, Regen, Schnee, Hagel, Sturm
- **Cluster 4**: Frankreich, USA, Russland, England, Kanada, Spanien
- **Singletons**: General, Meter, Baum, Junge, Dreieck, Durst, Dreck, Seil, Masern, Leder, Puppe, Taschentuch, Flagge, Rohr, Karneval, Gnade, Beule, Fass, Traum, Köder, Ordner, Magie, Handtuch, Uhr, Garten, Computer, Pflug, Distanz, Judo, Krankheit

A.2 Block Structure

Table A.1 shows the basic structure of a block. The codes used can be interpreted in the following way:

- The first character indicates the times of reinforcement. So ”1” stands for ”reinforced once” and ”2” stands for ”reinforced twice”. ”D” encodes distractor words that were never learned.
- The second character indicates whether the word is retrieved with or without a gap. ”0” stands for ”direct retrieval” and ”1” stands for ”delayed retrieval”. ”D” again encodes distractor words and ”F” encodes filler words that will never be retrieved.
- The third character indicates whether this is a cluster word or a singleton word. ”C” stands for ”cluster word” and ”S” stands for ”singleton word”.
- The fourth character is simply a counter that represents this element’s index within its class. So ”a” stands for ”first element”, ”b” for ”second element” and so on.

So for example ”10Ca” can be decoded as ”first (a) cluster (C) word that is reinforced once (1) and retrieved without a gap (0)”, whereas ”DDSc” can be decoded as ”third (c) singleton (S) distractor (D) word”.

Whereas the assignment of a word to the cluster property is given a priori (see section A.1), the assignment to the other properties (target, reinforcement, gap) and to the order within a set is done randomly.

A.3 Word List and Workload Mode Combinations

Table A.2 describes the combinations of word lists and workload modes for all participants. Participant IDs start at 4 because the IDs 1 to 3 have been used for a preliminary study.

Learn 1	Retrieve 1	Learn 2	Retrieve 2	Learn 3	Retrieve 3	Learn 4	Retrieve 4	Learn 5	Retrieve 5	Learn 6	Retrieve 6
10Ca	10Ca	10Cc	10Cc	11Cb	11Sb	11Sc	11Cb	11Cc	11Sc	10Sc	10Sc
10Sa	10Sa	11Sb	11Ca	20Sa	20Sa	20Sb	20Sb	20Cc	20Cc	20Sc	11Cc
10Cb	10Cb	20Ca	11Sa	20Cb	20Cb	20Cc	21Ca	20Sc	21Cb	1FSd	20Sc
10Sb	10Sb	20Sa	20Ca	20Sb	21Sa	21Cb	21Sb	21Cc	21Sc	1FSe	21Cc
11Ca	DDSa	20Cb	DDSc	21Ca	DDSe	21Sc	DDSg	1FSc	DDSi	1FSf	DDSk
11Sa	DDSb	21Sa	DDSd	21Sb	DDSf	21Cc	DDSh		DDSj		DDSl
20Ca	DDCa	21Ca	DDCc	21Cb	DDCe	1FSa	DDCg		DDCi		DDCk
21Sa	DDCb	21Sb	DDCd	21Sc	DDCf	1FSb	DDCh		DDCj		DDCl

Table A.1: This table shows the basic structure of a block.

Participant ID	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6
4	6 H	4 L	2 N	1 H	5 L	3 N
5	1 H	2 N	6 L	3 H	5 N	4 L
6	3 H	4 N	5 L	6 N	1 H	2 L
7	4 N	5 H	3 L	6 N	2 L	1 H
8	6 L	5 H	4 N	3 H	2 N	1 L
9	3 N	4 H	2 L	5 H	1 N	6 L
10	1 N	5 H	3 L	2 H	6 L	4 N
11	6 L	1 H	5 N	2 H	4 L	3 N
12	1 N	6 L	5 H	4 L	3 N	2 H
13	5 H	6 N	1 L	2 N	3 L	4 H
14	6 H	1 L	2 N	3 L	4 H	5 N
15	4 N	5 H	6 L	1 N	2 H	3 L
16	2 H	6 L	4 N	3 H	1 N	5 L
17	3 L	2 N	1 H	6 N	5 H	4 L
18	5 N	3 L	1 H	6 N	4 H	2 L
19	4 N	3 L	2 H	1 L	6 H	5 N
20	5 L	6 N	4 H	1 L	3 N	2 H
21	5 L	4 N	3 H	2 N	1 L	6 H
22	1 L	2 H	3 N	4 L	5 N	6 H
23	3 H	1 N	5 L	4 H	2 L	6 N
24	2 L	3 N	4 H	5 L	6 H	1 N
25	4 H	2 L	6 N	5 L	3 N	1 H
26	2 N	1 L	6 H	5 N	4 L	3 H
27	2 L	3 H	1 N	4 L	6 H	5 N

Table A.2: This table shows the combinations of word lists and workload modes for all participants. Each cell contains the number of the used word list followed by a letter indicating the workload mode. For example, ”6 H” in row ”4” and in column ”Block 1” means that word list 6 was used under workload mode HIGH in the first block for participant 4.

Reinforcement		
	reinforcement 1	reinforcement 2
Overall	1.02777 (0.58148)	0.96051 (0.64154)
None	0.94857 (0.49096)	0.82878 (0.33419)
Gap		
	gap 0	gap 1
Overall	0.95631 (0.66452)	1.03197 (0.55452)
None	0.84588 (0.37210)	0.93148 (0.46667)
Low	0.94776 (0.48950)	1.03638 (0.53913)
Target		
	distractor	target
Overall	1.03033 (0.54242)	0.99414 (0.61308)
None	0.96534 (0.47004)	0.88868 (0.42403)
Low	1.04170 (0.61170)	0.99207 (0.51698)
Correct		
	incorrect response	correct response
Overall	1.25493 (0.84462)	0.95666 (0.48194)
None	1.30584 (0.69007)	0.89413 (0.40571)
Low	1.21872 (0.66193)	0.96474 (0.52698)
High	1.2672 (0.99440)	1.02705 (0.50851)
Cluster (only distractor words)		
	cluster distractors	singleton distractors
Overall	1.06720 (0.59987)	0.99347 (0.47544)
None	1.01828 (0.52850)	0.91240 (0.39665)
High	1.11655 (0.56299)	1.05137 (0.49299)
Cluster		
	cluster	singleton
None	0.95441 (0.50567)	0.89969 (0.38271)

Table A.3: Table showing mean values and standard deviations (the latter noted in brackets) for all observed effects regarding response time. Response time is given in seconds.

A.4 Results

The tables in this chapter contain the mean values and standard deviations for all significant effects mentioned in section 3.2. Table A.3 shows the values for response time effects, table A.4 the values for response time effects when only considering correct responses. Table A.5 shows the values regarding hit rate effects and table A.6 the values regarding false alarm rate effects. Standard deviation is respectively noted in brackets behind the mean value.

Reinforcement		
	reinforcement 1	reinforcement 2
Overall	0.92747 (0.48183)	0.87689 (0.39910)
None	0.87880 (0.42435)	0.79618 (0.28762)
Gap		
	gap 0	gap 1
Overall	0.86660 (0.43416)	0.93930 (0.44198)
None	0.80073 (0.29012)	0.87313 (0.42249)
Low	0.86831 (0.44402)	0.94776 (0.42155)
High	0.95786 (0.55926)	1.03884 (0.47818)
Target		
	distractor	target
Overall	1.00067 (0.50834)	0.89981 (0.43916)
None	0.94761 (0.43574)	0.83524 (0.36099)
Low	1.01012 (0.58196)	0.90387 (0.43563)
High	1.04921 (0.49537)	0.99328 (0.52649)
Cluster (only distractor words)		
	cluster distractors	singleton distractors
Overall	1.03244 (0.56750)	0.97030 (0.44256)
None	0.98647 (0.47009)	0.91005 (0.39657)
Cluster		
	cluster	singleton
None	0.91251 (0.44602)	0.87589 (0.36057)

Table A.4: Table showing mean values and standard deviations (the latter noted in brackets) for all observed effects regarding response time for correct responses only. Response time is given in seconds.

Reinforcement		
	reinforcement 1	reinforcement 2
Overall	0.64352 (0.31430)	0.77662 (0.28055)
None	0.82813 (0.24114)	0.92361 (0.16344)
Low	0.59722 (0.30856)	0.76042 (0.29423)
High	0.50521 (0.29732)	0.64583 (0.29068)
Gap		
	gap 0	gap 1
Overall	0.77141 (0.27597)	0.64873 (0.32043)
None	0.91667 (0.16710)	0.83507 (0.24119)
Low	0.75000 (0.28741)	0.60764 (0.31996)
High	0.64757 (0.28576)	0.50347 (0.30123)
Cluster		
	cluster	singleton
Overall	0.74306 (0.28817)	0.67708 (0.31804)
Low	0.72569 (0.29538)	0.63194 (0.32171)
High	0.61806 (0.29334)	0.53299 (0.30520)

Table A.5: Table showing mean values and standard deviations (the latter noted in brackets) for all observed effects regarding hit rate.

Cluster		
	cluster	singleton
Overall	0.10359 (0.11959)	0.06192 (0.11226)
None	0.05208 (0.06345)	0.01910 (0.04934)
High	0.15625 (0.15243)	0.09028 (0.12845)

Table A.6: Table showing mean values and standard deviations (the latter noted in brackets) for all observed effects regarding false alarm rate.

B. Scripts and Programs

B.1 Experiment scripts

This section describes the scripts used for conducting the experiment.

B.1.1 Experiment Run Generator

The experiment run generator "ExperimentRunGenerator.py" is a python script programmed using Python 2.7.2 and the IDLE development environment. This script is used for generating the input files needed for the experiment script to conduct one experiment run.

The following input files are needed (all contained in the directory "raw_data/"):

- **id.txt**: Contains the next participant's id. IDs have to be in range [4,27] (IDs 1 to 3 have been used for a preliminary study and there are 24 participants). This file is automatically updated by the script.
- **configurations.txt**: Contains 24 experiment run configurations (one in each line) describing the order of word lists and workload modes. E.g. the configuration "6H,4L,2N,1H,5L,3N" means "word list 6 in workload mode HIGH, then word list 4 in workload mode LOW, then word list 2 in workload mode NONE, ..." (see also appendix A.3).
- **singletonsX.txt**: Contains the singleton words of word list X (with X in range [1,6]) – one word per line and 30 words in total.
- **clustersX.txt**: Contains the cluster words of word list X (with X in range [1,6]), again one word per line. Each of the cluster files contains 24 cluster words with six subsequent words being interpreted as a cluster (which means words 1 to 6 are interpreted as first cluster, words 7 to 12 as second cluster, and so on).

The script creates a mapping of the cluster and singleton words in the word lists using the given configurations. This mapping is then used to create the input files for the experiment script that contain the information of which word being learned and retrieved when and how.

Since the output of this script is assumed to be the input of the experiment script, it is written in the directory "input/".

B.1.2 Experiment Script

The experiment script "Workload Memory Task.psyexp" was designed using the PsychoPy 2 framework (version 1.73.06). See Peirce (2007) [24] and Peirce (2009) [25] for more information about this framework.

The experiment script needs the following input files in the folder "input/":

- **blockX/**: These folders (X in range [1,6]) contain the files generated by the experiment run generator. Each folder contains files "learnY.csv" and "trialY.csv" (Y in range [1,6]) containing the information about words to learn during the Y'th learning phase and words to retrieve during the Y'th retrieval phase, respectively. Moreover, a file "filenames.csv" is included which is used by the script to navigate through the folder.

- **overview.csv**: This file is used for navigation through the block folders and is also generated by the experiment run generator.

- **training_none/**, **training_low/** and **training_high/**: These folders contain the information needed for the training blocks. They are similarly structured as the "blockX/" directories but have only four learning and retrieval phases.

- **training_overview.csv**: This file is used for navigation through the training folders.

- **soundfiles.csv**: This file contains a list of file names. The referenced sound files contain the digits used for the secondary tasks.

In the directory "sounds/", there are ten sound files containing the digits used for the secondary task.

Before each run of the experiment script, the experiment run generator must be executed to generate a new experiment configuration for the new participant. When starting the experiment script, the experimenter is asked to provide participant name and session number.

The results of the experiment run will automatically be stored in the directory "data/". PsychoPy provides a plethora of files, but for further analyzing, only one file is of interest: "NAME_DATEtrials.csv" with NAME being the name entered as participant name and DATE being the date and time when the experiment run was started.

B.1.3 Result Converter

The result converter "ResultConverter.py" was also written using the PsychoPy 2 framework.

When started, a prompt asks the user to select files to convert for analyzing. These files must all be the "NAME_DATEtrials.csv" files mentioned above.

The result converter requires a directory "clusters/" containing files "cluster0.txt" to "cluster23.txt" which in turn contain the words belonging to a cluster.

The converter then computes hit rate, false alarm rate and accuracy and creates three csv output files: "rt_table.csv" (contains response times), "hit_table.csv" (contains hit rates) and "fa_acc_table.csv" (contains false alarm rates and accuracy).

These output files can then be analyzed, e.g. by using the statistical toolkit R.

B.2 Memory Model Program

The program "workload.jar" contains both the optimizing algorithm described in 5.1 and a simulator that uses the model to predict human memory performance.

B.2.1 Usage

The program can be started with the command line arguments listed in table B.1.

The bits of the bit masks correspond to the parameters "decay"-"intercept time"-"retrieval threshold"-"spreading potential"-"distractor factor"-"retrieval sensitivity". If the bit of a parameter is set to 1, this parameter will be optimized. If the corresponding bit is 0, the parameter will not be optimized but its default value will be used. Per default, all bits in the masks are set to 1.

Each digit of the filter masks can have one of the following values:

- "0": Collapse across this condition (i.e. transform the two condition values into one and compute overall mean).
- "1": Use only the first condition value.
- "2": Use only the second condition value.
- "3": Use both condition values.

The RT filter mask consists of four digits "cluster"-"target"-"reinforcement"-"gap", the hit filter mask consists of three digits "cluster"-"reinforcement"-"gap" and the false alarm mask consists of one digit "cluster".

For example, the RT filter mask "1033" means "consider only singletons, collapse over target and distractor words, and consider all different classes of reinforcement and gap", leaving the following classes: (cluster: singleton, reinforcement: 1, gap: 0), (cluster: singleton, reinforcement: 1, gap: 1), (cluster: singleton, reinforcement: 2, gap: 0), (cluster: singleton, reinforcement: 2, gap: 1)

Argument	Explanation
-sim	Only a simulation is run and no optimizing takes place.
-noSpreading	Deactivate spreading simulation.
-size	Population size for optimization algorithm, defaults to 100.
-it	Number of iterations for optimization algorithm, defaults to 1000.
-wRT	Weight of relative error of response time for fitness function, defaults to 1.
-wHit	Weight of relative error of hit rate for fitness function, defaults to 1.
-wFA	Weight of relative error of false alarm rate for fitness function, defaults to 1.
-maskNone	Bitmask enabling or disabling parameter optimization for workload mode None.
-maskLow	Bitmask enabling or disabling parameter optimization for workload mode Low.
-maskHigh	Bitmask enabling or disabling parameter optimization for workload mode High.
-filterRT	4 digit filter mask encoding the classes to use for optimizing the response time.
-filterHit	3 digit filter mask encoding the classes to use for optimizing the hit rate.
-filterFA	1 digit filter mask encoding the classes to use for optimizing the false alarm rate.

Table B.1: Table containing the command line arguments for the memory model program and their explanation.

As another example, the hit rate filter mask "302" means "consider both cluster and singleton words separately, collapse over reinforcement and consider only words that were retrieved with a gap", leaving the classes (cluster: cluster, gap: 1) and (cluster: singleton, gap: 1)

Per default, all digits of the masks are set to 3.

Consider the example in 5.2.1: To optimize only for the singleton target words in reinforcement and gap condition with spreading deactivated, a population size of 100, 1000 iterations and equal importance of response time and hit rate, the program should be called in the following way:

java -jar workload.jar -wFA 0 -maskNone 001001 -maskLow 101001 -maskHigh 101001 -filterRT 1133 -filterHit 133 -noSpreading

Since spreading is deactivated, it does not make sense trying to optimize the spreading potential, so the corresponding bit in all masks is set to zero. Moreover, the intercept time was defined to be 0.680 before, so its corresponding bit is also set to zero. Since the decay parameter was set to 0.5 for the NONE mode, also that bit is set to zero. Finally, f was kept constant zero, so the corresponding bit is not set.

For using the simulator, only two command line arguments are relevant: *-sim* and *-noSpreading*. *-sim* indicates that the simulator should be started and the optimizer will not be used, so it is always necessary when using the simulator. With *-noSpreading*, spreading can be deactivated. Since all other command line arguments have only an influence on the optimization algorithm, they are ignored in case *-sim* is set. When a simulation is run, the workload model parameters are taken from "defaultParams.csv" (see section B.2.2).

For example, to start a simulation with spreading activated, the program should be called in the following way: *java -jar workload.jar -sim*

Generally, it is advisable to increase the JVM heap size by using the JVM arguments "-Xms" (starting heap size) and "-Xmx" (maximum heap size). *"-Xms1024M -Xmx2048M"* has emerged as a good starting point.

B.2.2 Directory Structure and Needed Files

The memory model program needs to be run in a directory containing a subdirectory "resources/" that contains the files and folders listed in table B.2.

The csv files all have a defined structure which is described in the following paragraphs.

addedMemoryElements.csv has two columns: *id* and *string* with *id* containing a unique ID for the new memory element (must not be already in use by WordNet) and *string* containing a string representation of the new memory element.

addedPointers.csv has two columns: *parent_id* and *child_id*. Each row represents a bidirectional edge between two synsets which are named by their synsetID.

defaultParameters.csv has six columns. Each row is interpreted as parameter configuration for one workload mode (in order NONE, LOW, HIGH).

- *decay*: the decay parameter d.

Directory / file name	Description
addedMemoryElements.csv	Contains a list of memory elements added to the WordNet database.
addedPointers.csv	Contains the pointers added to the WordNet database.
defaultParameters.csv	Contains the default parameter configurations for the three workload modes. Those will be used for a simulation run and as a default value for optimization (if the corresponding bit in the bit mask is set to zero).
dict/	Contains the WordNet dictionary information. This is simply the "dict/" directory of any WordNet distribution.
dictionary.csv	Provides a translation between the word strings (as used in the experiment) and corresponding WordNet synset IDs.
ExperimentRunGenerator.py	Python script for generating the data for a new experiment run that will be stored in the "input/" folder. See section B.1.
input/	Contains the information of the current experiment run. Will be updated by ExperimentRunGenerator.py. See also section B.1. Contains also the actual experiment's results.
parameterRanges.csv	Defines the legal ranges for the parameters. This will be used by the optimizer to systematically search for legal solutions.
raw_data/	Contains the word lists. See section B.1.

Table B.2: Table showing the files and subdirectories needed to run the memory model program.

- *intercept*: the intercept time I.
- *retrieval_threshold*: the retrieval threshold τ.
- *spreading_potential*: the spreading potential P_{spread}.
- *distractor_factor*: the distractor time factor f.
- *retrieval_sensitivity*: the retrieval sensitivity s.

dictionary.csv has two columns: *word* and *id*. Each row is interpreted as a tuple and defines a mapping between a word and the ID of the corresponding synset.

parameterRanges.csv has 12 columns and should contain only one row of data. The columns are *decay_min*, *decay_max*, *intercept_min*, *intercept_max*, *thresh_min*, *thresh_max*, *spread_min*, *spread_max*, *distr_min*, *distr_max,* , *sensitivity_min* and *sensitivity_max,* . They define the legal values for the parameters listed in defaultParameters.csv by defining a minimum and a maximum value.

The directory **input/** contains the information for the current experiment run, but it also contains the experiment results used for optimizing the model. They are located in the subdirectory **input/experiment_results/**. The file **filenames.csv** has three columns *rt_file*, *hit_file* and *fa_file* that contain the relative path for the files containing response times, hit rates and false alarm rates for the workload stages. Again, each row is interpreted as one workload mode (in order NONE, LOW, HIGH).

The referenced files themselves contain only a single column called *mean*. In this column, the mean values for the different subsets are listed. Table B.3 contains a table showing the order of the subsets within the mean column. The codes can be interpreted in the following way:

- **S** stands for "singleton", whereas **C** stands for "cluster".
- **T** stands for "target word", whereas **D** stands for "distractor word".
- **R1** stands for "reinforced once" and **R2** stands for "reinforced twice".
- **G0** stands for "retrieved without a gap" and **G1** stands for "retrieved with a gap".

So for example, "S-T-R2-G0" is the subset of singleton (S) target words (T) that have been reinforced twice (R2) and that are retrieved without a gap (G0).

Once the program is started, it will print a log file **output/log.txt**. Moreover, the simulation results are saved in **output/stats.csv**. The latter file has only one column *mean* which contains the mean values for the different variables in the different sets. The mean values are in the same order as shown in table B.3. The different variables are listed in the following order: response time, hit rate, false alarm rate. They are separated by the string "newTargetValue". First, all variable means of workload mode NONE are listed, then all of workload mode LOW and then all of workload mode HIGH. Workload modes are separated by the string "NEW_WORKLOAD".

Moreover, there is a spreadsheet **GraphGenerationSpreadsheet.ods** which can be used to analyze the simulation results and to obtain the graphs used in chapter 5 by simply pasting the content of stats.csv for different evaluation steps.

response time	hit rate	false alarm rate
S-T-R1-G0	S-R1-G0	S
S-T-R1-G1	S-R1-G1	C
S-T-R2-G0	S-R2-G0	
S-T-R2-G1	S-R2-G1	
S-D	C-R1-G0	
S-D	C-R1-G1	
S-D	C-R2-G0	
S-D	C-R2-G1	
C-T-R1-G0		
C-T-R1-G1		
C-T-R2-G0		
C-T-R2-G1		
C-D		
C-D		
C-D		
C-D		

Table B.3: This table shows the order of the subsets.

Bibliography

[1] Sandra G. Hart and Lowell E. Staveland. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, page 39–183. 1988.

[2] Jr. Lyle E. Bourne and Rita A. Yaroush. Stress and Cognition: A Cognitive Psychological Perspective. Technical report, National Aeronautics and Space Administration, 2003.

[3] Fergus I. M. Craik, Moshe Naveh-Benjamin, Richard Govoni, and Nicole D. Anderson. The Effects of Divided Attention on Encoding and Retrieval Processes in Human Memory. *Journal of Experimental Psychology*, 125(2):159–180, 1996.

[4] Holger Schultheis, Thomas Barkowsky, and Sven Bertel. LTM^C - An Improved Long-Term Memory for Cognitive Architectures, 2006.

[5] 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.

[6] Dominic Heger, Felix Putze, and Tanja Schultz. Online Workload Recognition from EEG data during Cognitive Tests and Human-Computer Interaction. In *33rd Annual German Conference on Artificial Intelligence 2010*, 2010.

[7] Felix Putze and Tanja Schultz. Towards Cognitive Dialog Systems. In *1. Fachtagung Biophysiologische Interfaces*, 2009.

[8] Allan Newell. *Unified Theories of Cognition*. Harvard University Press, 1990.

[9] Nelson Cowan. Evolving Conceptions of Memory Storage, Selective Attention, and Their Mutual Constraints Within the Human Information-Processing System. *Psychological Bulletin*, 104(2):163–191, 1988.

[10] Larry R. Squire. Declarative and nondeclarative memory: Multiple brain systems supporting learning and memory. *Journal of Cognitive Neuroscience*, 4(3):232–243, 1992.

[11] Bruno Emond. WN-LEXICAL: An ACT-R module built from the WordNet lexical database, 2006.

[12] Dan Bothell. *ACT-R 6.0 Reference Manual*.

[13] ACT-R 6.0 Tutorial. http://act-r.psy.cmu.edu/actr6/units.zip, Dec 2011.

[14] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Introduction to WordNet: An Online Lexical Database, 1993.

[15] George A. Miller. Nouns in WordNet: A Lexical Inheritance System, 1993.

[16] Dean C. Delis, Paul J. Massman, Edith Kaplan, Richard Mckee, Joel H. Kramer, and Dennis Gettman. Alternate Form of the California Verbal Learning Test: Development and Reliability. *The Clinical Neuropsychologist*, 5(2):154–162, 1991.

[17] Jason Brandt. The Hopkins Verbal Learning Test: Development of a New Memory Test with Six Equivalent Forms. *The Clinical Neuropsychologist*, 5(2):125–142, 1991.

[18] William F. Battig and William E. Montague. Category Norms for Verbal Items in 56 Categories: A Replication and Extension of the Conneticut Category Norms. *Journal of Experimental Psychology Monograph*, 80(3):1–46, June 1969.

[19] Kenneth Craik. *The Nature of Explanation*. Cambridge University Press, 1943.

[20] Christopher D. Wickens. Processing resources in attention. In *Varieties of attention*, pages 63–102. 1984.

[21] Myra A. Fernandes and Morris Moscovitch. Divided Attention and Memory: Evidence of Substantial Interference Effects at Retrieval and Encoding. *Journal of Experimental Psychology*, 129(2):155–176, 2000.

[22] Robert H. Logie, Sergio Della Sala, Sarah E. MacPherson, and Janine Cooper. Dual Task Demands on Encoding and Retrieval Processes: Evidence from Healthy Adult Ageing. *Cortex*, (43):159–169, 2007.

[23] Stephen Monsell. Task switching. *RENDS in Cognitive Sciences*, 2003.

[24] Jonathan W. Peirce. PsychoPy - Psychophysics Software in Python. *Journal of Neuroscience Method*, (162):8–13, 2007.

[25] Jonathan W. Peirce. Generating stimuli for neuroscience using PsychoPy. *Frontiers in Neuroinformatics*, 2, 2009.

[26] Stephan Lewandowsky and Klaus Oberauer. No Evidence for Temporal Decay in Working Memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 35(6):1545—-1551, 2009.

[27] Robert Pröpper and Felix Putze. Adaption of cognitive models to dynamically changing mental workload. Master's thesis, Karlsruhe Institue of Technology, 2011.

[28] Kurt Mehlhorn and Peter Sanders. *Algroithms and Data Structures - The Basic Toolbox*, chapter 12.6 Evolutionary Algorithms, pages 259–261. Springer-Verlag Berlin Heidelberg, 1st edition, 2008.

[29] Stuart Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach*, chapter 4.1.4 Genetic Algorithms, pages 126–129. Pearson Education, 3rd edition, 2010.