MASTER THESIS

Optimization of a control network for binding in combinatorial sentence structures.

Koen Meijer, s1744275

Faculty of Cognitive Psychology and Ergonomics
Human Factors and Engineering Psychology

EXAMINATION COMMITTEE
Prof. F. van der Velde & Dr. M. Schmettow.

1-8-2019

UNIVERSITY OF TWENTE.

Optimization of a control network for binding in combinatorial sentence structures.

Koen Meijer, s1744275
Date: 1-8-2019
Supervisors: Prof. F. van der Velde & Dr. M. Schmettow.
Human Factors and Engineering Psychology
Department of Cognitive Psychology and Ergonomics
University of Twente, Enschede

Abstract

The ability of human to understand and produce combinatorial structures in language is an important feat of cognition. It allows for the understanding of a huge set of sentences generated by the combination of a known sentence structure and known words. How these combinatorial structures can be instantiated in neural terms is still faced by some challenges. The neural blackboard architecture (NBA) aims to solve these challenges. In this architecture words are bound together in a temporary sentence structure that encodes the relations between the words. The question remains with what network the binding in the NBA can best be controlled. Therefore, the goal of this thesis was twofold. The first being the replication of the FFN trained by van der Velde and de Kamps (2010). The second being finding the best hidden layer size for this FFN. Accuracy as reported by Keras showed a decline for processing of the test sentences for the models with six or fewer hidden nodes. However, a model configuration that completely omitted the hidden layer scored an accuracy equal to the model with 12 hidden nodes. Further inspection of the activations made by the models revealed that the model without a hidden layer made more errors. In conclusion it can be said that a model without a hidden layer is not adequate to perform the controlling task for binding in the NBA if the performance requirements of van der Velde and de Kamps (2010) are applied. However, when the standard Keras thresholds for accuracy are used the model without a hidden layer seems to be able to control the binding. This points towards the data used for this control task being linear separable.

Contents

1. Introduction

Language comprehension or semantic cognition is the ability to use, manipulate and generalize knowledge to support both verbal and non-verbal behaviours (Lambon-Ralph, Jefferies, Patterson & Rogers, 2017). A fundamental characteristic of human semantic cognition is that it relies on combinatorial structures. Humans use combinatorial structures in language by combining vowels and consonants into syllables, syllables into words and words into sentences (Zuidema & de Boer, 2018). This ability is almost unending as humans can always create new words using known phonemes and can express complex thoughts using new combinations of words forming novel sentences. This combinatorial productivity even results from combining simple sentence structures with a large but finite lexicon. It is not unlimited but still results in a huge set of sentences that can be understood because this results from combining a finite lexicon with restricted sentence structure. When new words are learned in a particular sentence context, these words can also be recognized in other sentence contexts without explicitly learning these first. Adult language users possess this form of productivity. Thus, with combinatorial productivity new words only have to be learned in a few sentence contexts. They could then be used in other sentence context as well..

Studying how these combinatorial structures can be instantiated in neural terms is essential to understanding the basis of human cognition. The representation of words and concepts are formed by neural cell assemblies (Hebb, 1949). Such an assembly consists of a group of interconnected neurons that is distributed across the brain. The hub and spokes model of semantic cognition (Lambon-Ralph et al., 2017) is a more recent version of this hypothesis. In this model semantic cognition relies on two interacting neural systems. The first system is a system of representation in which knowledge of concepts is encoded. The second system manipulates the activation within the first system and is therefore a system of control. This two systems view is called the controlled semantic cognition (CSC) framework (Lambon-Ralph et al., 2017). However, the neural instantiation of the combination of such structures poses some fundamental problems. These problems are the massiveness of binding, the problem of two, the problem of variables and the transformation of combinatorial structures from working memory to long-term memory (Jackendoff, 2002). The Neural Blackboard Architecture (NBA) (van der Velde & de Kamps, 2006) can solve these challenges to cognitive neuroscience. Moreover, this is a neural architecture that aims to integrates three important features of human cognition. These features are productivity, dynamics and grounding. Productivity refers to the combinatorial nature of cognition as

described above. Dynamics refers to the ability of interacting with the environment in a dynamical way. Grounding refers to the nature of cognitive representations.
These representations are always grounded in perception. Especially the combination of these features is important to human cognition and not found often, except in the human brain. Therefore, to better understand the nature of human cognition an architecture that integrates these features is important (van der Velde & de Kamps, 2006). The NBA can create combinatorial structures. However, without a control network that has learned in which way to activate certain structure assemblies the NBA is not capable of processing novel structures.

A control network is therefore needed to make the model capable of productivity. Once a specific sentence structure is learned; let's say a N-V-N structure, this can be generalized to all sentences of this type as long as it consists of nouns and verbs that are known to the human/model. So the need to train a neural network on a significant number of sentences of the overall set humans can understand can be avoided (van der Velde, van der Voort van der Kleij & de Kamps, 2004). In this study a control network for a model of sentence structure will be tested. This control network will provide the necessary input for the NBA (van der Velde & de Kamps, 2006) to make it capable of processing novel sentences. Further testing with the control network will determine whether it is capable of processing the combinatorial structures in the same way as humans. A control network could make the NBA capable of processing novel sentences just like humans are able to process and produce combinatorial structures (van der Velde & de Kamps, 2006). The first part of this of thesis will be dedicated to replicating the study by van der Velde and de Kamps (2010) on learning of control in a neural architecture of grounded language processing. The second part will consider trying different configurations of the model to optimize the performance. Mainly, the effect of the hidden layer size in the neural network will be studied.

2. Theoretical framework

2.1 Combinatorial productivity/structures.

The combinatorial nature of language refers to the way human combine sounds to form word (combinatorial phonology) and the way in which humans combine words into sentences (compositional structures) (Zuidema & de Boer, 2018). In this way humans can combine words to form novel sentences, as long as the words are known.

The ability to provide 'who does what to whom' information can be seen as the communicative aspect of language. Two sentences with the same syntactic structure can give very different information. By answering who does what to whom question this information can be derived. This is called a binding question since it is related to the manner in which the constituents in the sentence structure are bound together. For instance, the information in a sentence changes meaning when a noun goes from the object to the subject position. Answering such binding questions is an important feature of human language processing (van der Velde, van der Voort van der Kleij & de Kamps, 2004).

Human language processing is very productive in representing who does what to whom information. With a vocabulary of 1000 words and one specific sentence structure a huge set of sentences can be created. Every human that knows the words that are in the vocabulary and is familiar with the specific sentence structure can then understand each sentence in this set and would be able to answer binding questions. The combinatorial productivity of language results from combining sentence structures with a large lexicon (vocabulary). This will result in a set of sentence which is 'astronomical' in its size but still understandable to the average person because they are the result of combing a large but finite lexicon with restricted sentence structures (van der Velde, van der Voort van der Kleij & de Kamps, 2004). To deal with learning a set of this size combinatorial productivity in a model is also needed. This way, not every combination of words and sentence structures has to be trained but new words can be trained in a few contexts and then used in others. Or a model can learn different sentence structures without simultaneously learning specific words.

The ability of a model to learn a specific word in one type of sentence and then use it in another is a form of productivity referred to as strong systematicity. However, this type of systematicity could be difficult for a neural network to learn because the word as well as the syntactic position have to be learned at the same time. Moreover, for combinatorial productivity strong systematicity may be not be necessary. Alternatively, a form of weak systematicity, in which each word is trained on all possible syntactic positions in the new test

sentences may be sufficient to produce combinatorial productivity (van der Velde, van der Voort van der Kleij & de Kamps, 2004).

2.2 Grounded representations

A grounded representation of a specific word is made up from all the related aspects to that word in an interconnected network. The aspects that are related include perceptual information, processes related to actions (petting a cat), emotional information related to the word and all other forms of information.

Several studies have investigated grounded word representations in the brain. One interesting finding is that of modality specific activation in the brain during the comprehension of words. Motor related words activated brain areas that are associated with motor skills and sensory brain areas where activated during the comprehension of sensory related words (Vigliocco et al., 2006). Difference in activation between several words related to motor action were also found. Parts of the premotor cortex that are activated by specific actions were activated by words related to these specific actions (Tettamanti et al., 2005).

The hub-and-spoke theory of semantic representations explains how conceptual knowledge can arise. This theory assumes that the most important sources for constructing concepts are multimodal verbal and non- verbal experiences. These sources of information are encoded in modality specific brain areas that are distributed across the brain. These are called the *spokes*. The second idea of this theory is that interactions across modalities are mediated by a trans-modal *hub* that is situated bilaterally in the anterior temporal lobes (ATL) (Lambon-Ralph et al., 2017).

There is empirical evidence that motivates that the hub is situated on the anterior temporal lobes, mainly from cognitive neuropsychological studies. Damage to higher-order association cortices can lead to trans-modal semantic impairments. An interesting study on semantic dementia (SD) showed that the ATL-hub might be important for all types of concepts. Patients with SD show atrophy and hypometabolism in the anterior temporal regions bilaterally. With semantic SD there are impairments across all modalities and all types of concepts. This seems to point towards a central, trans-modal hub (Lambon-Ralph et al., 2017).

2.3 Recurrent Neural Networks and combinatorial productivity

Recurrent neural networks can be used to study sentence processing. An example of such a study is the word prediction task with a simple recurrent neural network (SRN)

(Elman, 1991). A SRN is essentially a multilayer neural network with connections between the input layer and hidden layer and the hidden layer and output layer. Recurrence is made possible by copying information from the hidden layer back to the input layer. This is done via the context nodes as shown in figure 1. The information that is stored in the context layer is combined with the next input. In this way the hidden layer of the model relates both new and prior information to the output layer (Elman, 1991). This model was trained on a set of sentences so that it could predict the lexical category of the next word in a new sentence.

In a more recent study a SRN was used to examine whether such a model would be capable of combinatorial productivity (van der Velde, van der Voort van der Kleij & de Kamps, 2004). However, the model that was used failed on the test of combinatorial productivity. The SRN was capable of learning the training sentences but with the test sentences in which words from the learned sentences were combined it failed to correctly predict the lexical category of the next word. In their article they call this a falsification of the idea that a SRN can manage human language productivity (van der Velde & de Kamps, 2004).

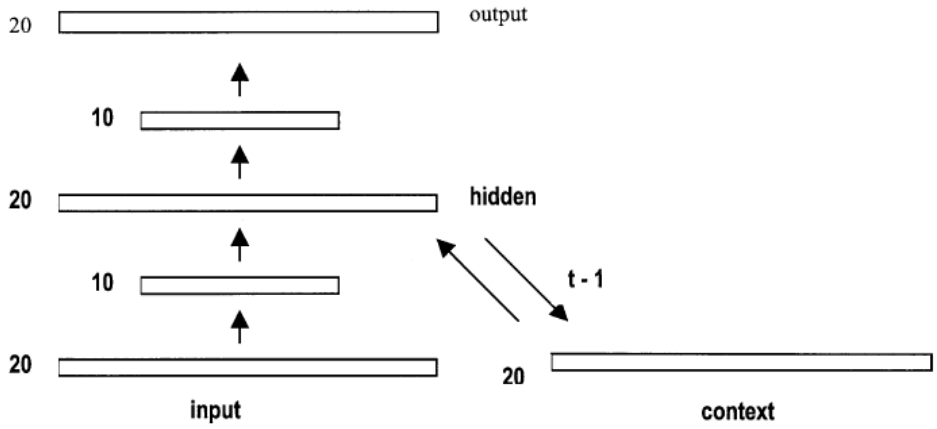

Figure 1: A simple recurrent neural network (van der Velde, van der Voort van der Kleij & de Kamps, 2004).

Combinatorial productivity is not the same as recursive productivity. Recursive productivity refers to processing increasingly complex structures whereas combinatorial productivity concerns combining a big lexicon with limited syntactic structures (van der Velde & de Kamps, 2006). The long short-term memory recurrent network has, as opposed to SRN and humans, unlimited recursive productivity. It can also handle context free language by using ‘memory units’ that work as counters during learning. However, counting elements in a sentence is not useful for natural language. By simply counting how often a specific word

is encountered in a sentence differences between sentences cannot be represented. Additionally, this method would not work for new sentences with words that have not been encountered before and therefore do not have a counter yet. This shows that the ability to process increasingly complex languages does not assure that combinatorial productivity can also be processed (van der Velde & de Kamps, 2006).

Another issue with a SRN is related to the binding problem, which entails the way in which the structural relations between parts can be kept when these parts are bound together momentarily. Instead of processing words directly a SRN could be trained to process sentences based on their abstract structure. The problem with this however would be that all sentences with the same syntactic structure, such as Noun-Verb-Noun would be the same for this model even though they convey different messages. Humans have the ability to answer binding questions for this type of sentence but these questions are impossible to answer based solely on the abstract sentence structure. To be able to distinguish the different messages of sentence with a similar abstract structure the binding has to be performed according to the hierarchical structure of the sentence. This is a form of language processing of which a SRN is not capable (van der Velde & de Kamps, 2006).

2.4 The Neural Blackboard architecture of combinatorial structures.

In a combinatorial structure parts and their relations are both present. This means that a neural model of combinatorial structures can only be useful when it instantiates the individual parts as well as the relations between them in a combinatorial structure. Moreover, in a sentence structure the individual parts consist of the grounded word representations as described in the hub-and spoke model (Lambon-Ralph et al., 2017). The combination of combinatorial productivity with grounded word representations presents challenges. It means that connections should be possible between all of the different grounded representations of nouns and all grounded representations of verbs. Additionally, connections between these grounded representations cannot be formed directly when a new combination of nouns and verbs is encountered. Therefore, combinatorial productivity has to be possible in a connection structure that is already established (van der Velde, 2019).

The neural blackboard architecture makes this combination possible by forming a temporary connection between arbitrary nouns and verbs (van der Velde & de Kamps, 2006). This temporary connection structure is formed on a ‘blackboard’ consisting of specialized processors that can interact. Each of these processors can process and alter the information on the blackboard. The processors in the NBA are called structure assemblies which are linked to

the grounded word representations in a sentence structure. There are *noun assemblies* and *verb assemblies*. When a sentences is processed all the nouns are linked to an individual noun assembly and the verbs are linked to verb assemblies. The noun and verb assemblies are linked together in a manner that encodes the relations between these words. Within the structure assemblies for nouns and verbs are subassemblies for the different roles these words can have in a sentence structure. For example, for the sentence *dog sees cat*, *dog* and *cat* are linked to two different noun assemblies. *Sees* is linked to a verb assembly. Next, these assemblies are linked together by the appropriate subassemblies. *Dog* is linked as the *agent* of *sees,* and *cat* as the *theme* of *sees*. This linking of structure assemblies in the NBA is called binding. The binding in the architecture is achieved by activating the correct binding gates in the structure assemblies (for a detailed description of binding gates in the NBA sees van der Velde & de Kamps, 2006).

In the NBA the activation of binding gates needs to be controlled. A control network for this task was proposed by van der Velde and de Kamps (2010). This network received information about the word types in a sentence structure and feedback from the NBA concerning expectations of the sentence structure. For instance, verbs usually have themes therefore when a theme is encountered this is signalled to the control network. How this control works is discussed in detail in the next section.

2.5 Control of binding in the Neural Blackboard Architecture

In a recent study a control network for the neural blackboard architecture was presented (van der Velde & de Kamps, 2010). The network that was used to control the binding was a feedforward network (FFN) with one hidden layer that operated on abstract binding symbols. These binding symbols/categories could then be used to activate the appropriate structure assemblies in the NBA. The main goal of this study was to investigate whether a neural network is capable of generalizing knowledge of these binding classes to novel sentence structures. To this intent two main types of sentences were trained. Namely, sentences with a noun-verb-noun structure and sentences with an embedded clause. Below is a description of the input-output relations the FFN learned to control the binding.

During binding, nodes are activated that represent a lexical category. This is done from the assumption that grounded word representations also entail information about lexical categories. For the noun-verb-noun sentences there are several of these categories that were uses to train the control network on the binding steps. These are nouns with a relative clause (Nrc), nouns with a complement clause (Ncc), complementizers (C) and verbs (V). In addition

there are two categories that are meant for starting and ending sentence structures in the NBA and there are conditional nodes. Figure 4 shows the binding steps for the sentence *cat sees dog*. For each word in this sentence ate least one or more nodes in the input- and output layer are activated. All the necessary steps in the binding process for this sentences is discussed below.

The FFN that was trained operates on sentence level in which the parts of the sentence structure are processed word for word. The first action is that *cat* activates an input node in the FFN. This does not happen directly but rather by recognizing the category to which *cat* belongs first. In the case of *cat* this category is noun with a possible relative clause (Nrc). In the input layer of the FFN there is a node dedicated for signalling the to the output layer that the input is Nrc. The trained model has learned to activate the correct output in this case. The correct output that needs to be produced by the FFN consists of two nodes. The first is for activating the noun *cat* as the subject of the sentence (N-n-S). This node will signal to the NBA that these two structure assemblies are bound together via their noun-subassemblies. *Cat* (N1) will then be bound to the sentence structure (S1) as the subject. Additionally, the FFN has learned to activate a node that reflects the expectation of an incoming relative clause (RC). This is an example of a conditional node that is not used as input for the model directly. Rather, it is used as feedback for the FFN. The RC node will serve as input together with a new input when it meets the condition of being a relative clause. In the case of *cat sees dog* this condition is not met since there is no incoming relative clause after *cat* (van der Velde & de Kamps, 2010).

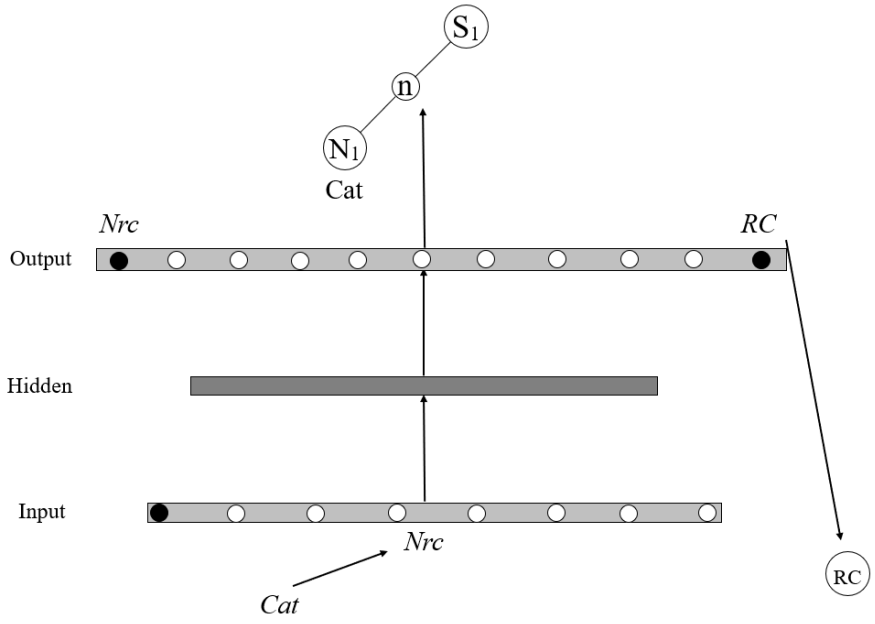

Figure 2: Schematic presentation of the processing of a noun with possible relative clause *Cat* through the FFN. Dots represent nodes in the input- and output layer; black dots are nodes that are activated.

The verb *sees* falls under the verb category which activates the V node in the input layer of the FFN as displayed in figure 3. The trained FFN has learned to activate two nodes for verb inputs. The first is the V-v-S node. When this node is activated it signals to the NBA that the verb (V1) should be bound to the sentence structure (S1) as a verb. Since verbs usually have a theme an additional node is activated in the output layer of the FFN. The V-t node reflects the expectation that the next word in the sentence structure could be the theme of the verb *sees.* In the NBA, the result is that a gate is opened for theme-subassembly in the V1 structure assembly. A feedback signal is also produced. The conditional node T is activated and will remain active until new input will satisfy the condition of being a theme (van der Velde & de Kamps, 2010).

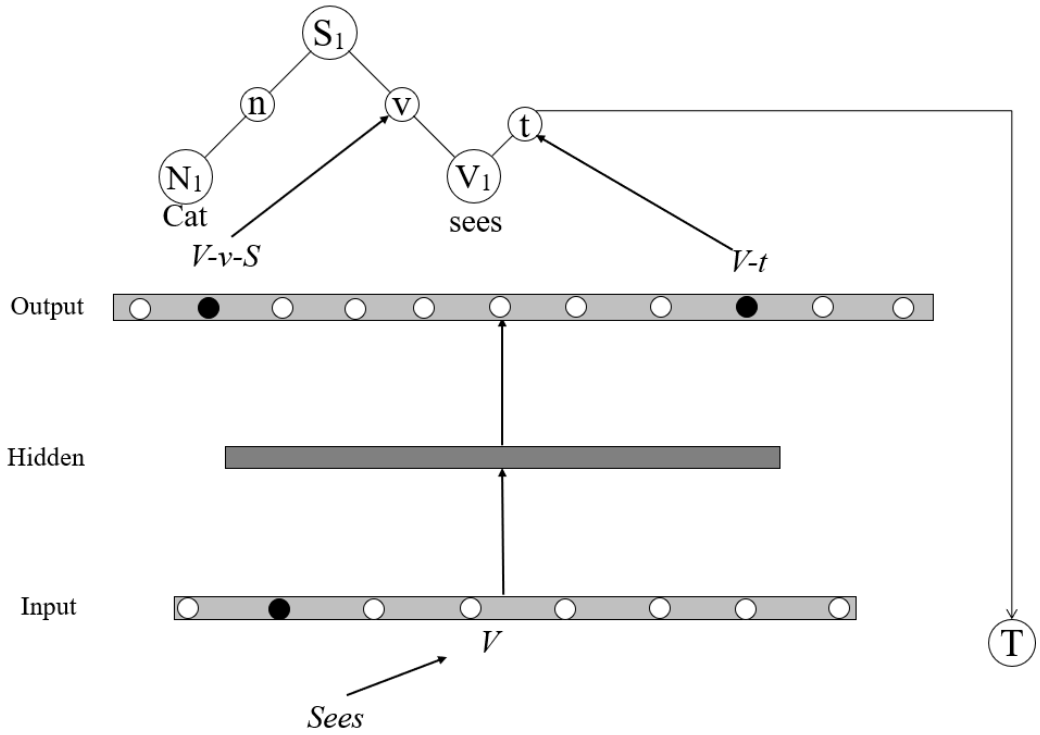

Figure 3: Binding of the verb *sees* as the verb of the sentence structure *cat sees dog*. The anticipation of a theme is reflected by the V-t node and the conditional T node which is given as feedback by the NBA.

The last word of the sentence, *dog*, is another noun with a possible relative clause (Nrc). Therefore, similarly to the first noun *cat* the Nrc node in the input layer of the FFN is activated. However, this time another node in the input layer is active. The conditional T node that was given as feedback after processing the verb *sees* is also active. The FFN has learned to activate two nodes in this case. The RC node for a possible incoming relative clause and the N-t node. This N-t node signals that the noun *dog* is a theme and therefore the theme subassembly of N2 should be activated. Together with the theme subassembly of V1 this forms the binding of *dog* as the theme of the verb *sees*. Since the condition of the T conditional node is now met it is no longer activated and therefore not given as feedback again (van der Velde & de Kamps, 2010).

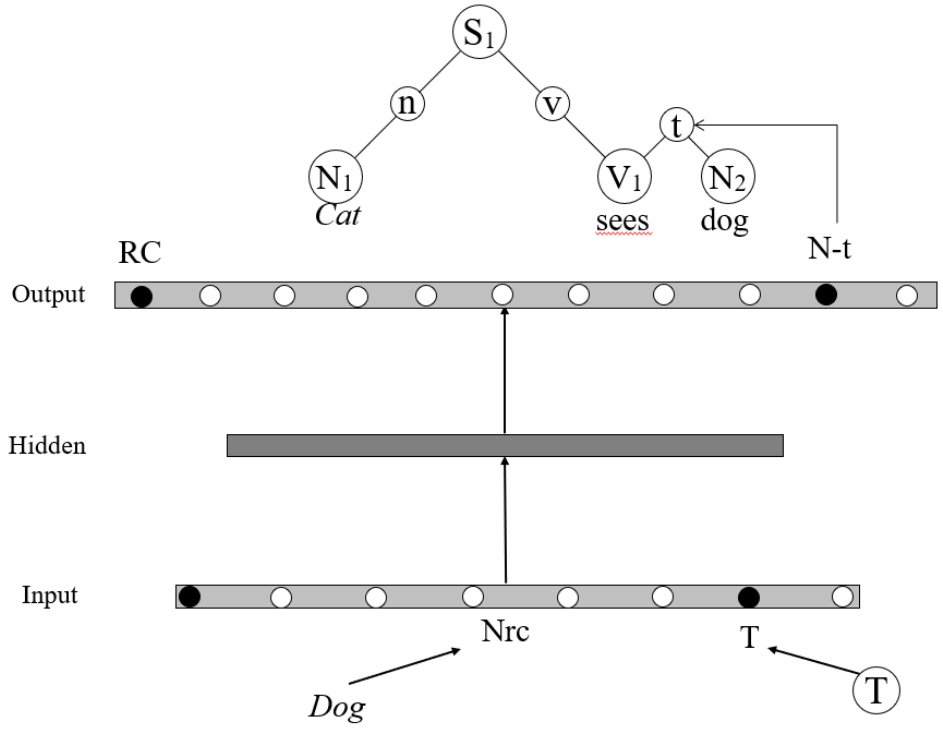

Figure 4: Activation in the FFN for the noun *dog* in the sentence *cat sees dog*. The result is that *dog* is bound as the theme of the verb *sees.* Black circles represent activated nodes in the in-and output layer.

2.6 Research aims

The first aim of this study was to replicate the results by van der Velde and de Kamps (2010). A Feedforward neural network with 12 hidden nodes was trained on noun-verb-noun

sentences and sentences with an embedded clause. After training the model was evaluated to see if it is capable of controlling the binding task similarly to the original model.

After the replication of model the second aim was to evaluate the effect of hidden the layer size on the performance of the model. In the original study the number of hidden nodes was chosen without explicit thought. Therefore, the hidden layer size was lowered from the original size of 12 nodes until a model was created without a hidden later at all, called a perceptron. All of these architectures will be discussed in the sections below.

In the original study the number of epochs for training was set to 50.000. However, this number was chosen without a specific reason. Therefore, during the evaluation the number of epochs and the extent to which training for 50.000 epochs is necessary for the binding task was also evaluated.

The last research aim is related to the tools that were used in the original study. At that time software packages such as Keras were not available for use. Keras provides a clear framework for creating neural networks and makes it more accessible to be used by those who have no training or expertise in computer science or machine learning.

3. Network architectures

A Feedforward Neural Network for classification was used to serve as a control structure for the Blackboard architecture. The code for this model can be found in appendix 1. This type of network is also called a Multilayer Perceptron and is illustrated with figure five. It consists of three densely connected layers of which the middle is the 'hidden layer'. The first layer is the input layer and contains eleven nodes or *neurons.* Each neuron in the input layer is connected to each neuron in the hidden layer. In turn each neuron in the hidden layer is connected to the neurons in the output layer that consist of 14 nodes. In this way the neurons can take the input data or the output of other neurons as input. These data are then multiplied by the neurons *weight*. This is initially a random number but through optimization the weight is adjusted so that the prediction of the model becomes closer to the desired value. This is done by calculating a distance value to the true score, the *loss value.* The gradient of the loss is calculated and the parameters of the network are moved in the opposite direction of the gradient (Chollet, 2018). This way the combination of weights in the network that creates the smallest loss value can be found. These computations involve a series of computations to calculate the gradient, called *backpropagation.* The process of optimizing the network based on these computations is called *stochastic gradient descent* (Chollet, 2018).

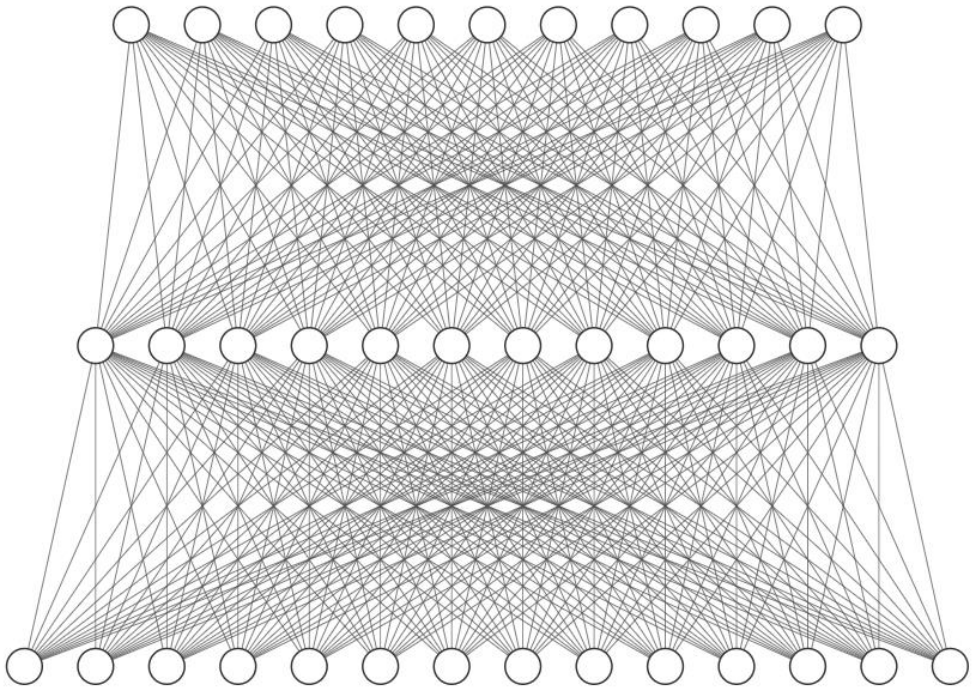

Figure 5: Multilayer perceptron/Feedforward neural network with one hidden layer consisting of 12 nodes.

As the activation function in the last layer of the network the Sigmoid function was used. The sigmoid function is defined as:

$$f(x) = \frac{1}{1 - e^{-x}}$$

The sigmoid function has a distinctive s-shape as shown in figure six. This shows that the sigmoid function is a 'squashing function'. A large range of values are squashed into the range of 0 and 1. This value indicates the probability that a given input belongs to a certain output class. Since this value is calculated for each neuron in the output layer separately this function can handle the data used to train the model. The classification task that needs to be performed to control the binding in the blackboard architecture is one in which each input can be related to multiple output categories simultaneously. This is called a multi-label classification task.

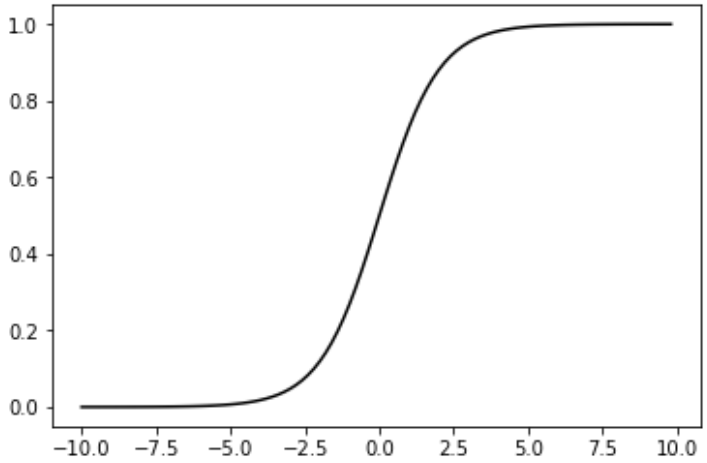

Figure 6: The sigmoid function

All of the FFN models that were used in this study only differ with respect to the size of the hidden layer. The middle section of the connections structure in figure five becomes smaller as the number of nodes decreases. This also decreases the number of connections that are made between the three layers. The final model that was tested consisted of only two layers, an input and an output layer. This connection structure is pictured in figure 11. Such a connection structure in which the input directly passes through to the output layer is called a perceptron model. The process through which a perceptron model learns is different compared to that of the FFN model described above. A perceptron model does not have a hidden layer of neurons. It can only handle data that can be separated in a linear manner. This is directly related to the classification task a model needs to perform. There are two forms of classification that are linearly separable and can be solved by a perceptron model. These are

AND and OR classification. These two types of classification will be illustrated with a perceptron model consisting of two layers illustrated in figure seven. In the first layer there are two neurons; x and y. The last layer consists of 1 neuron which is the output of the model. All of the neuron can have a value of 1 or 0.

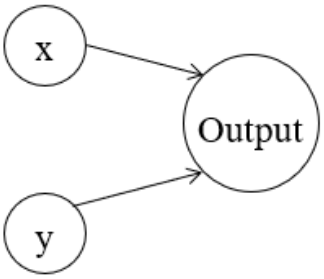

Figure 7: Simple perceptron with two input nodes: x and y.

For the AND classification the neuron in the output layer will have a value of 1 if both the x and y neuron have a value of 1. This gives the following input space as shown in figure 8. In table 1 the result of the different combination of x and y on with the AND classification are given. These points can be plotted as x,y coordinates. In figure eight this is translated to a dot for the output value being 0 and triangle for an output value of 1. The red line shows that for this input space, a linear line can separate the dots from the triangle.

Table 1.

AND classification

x	y	AND
1	1	1
0	1	0
1	0	0
0	0	0

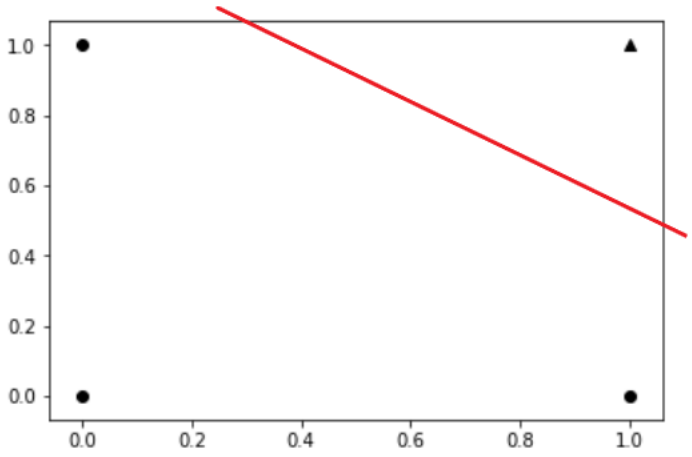

Figure 8: Input space for AND classification. A single line can separate the two classes: dots and triangles.

The second example of classification that a perceptron model can handle is OR classification. In this case the output of the model becomes 1 when either the x or y node is a 1. This gives the following output as sees in table 2. In contrast to the AND classification this type of classification results in more values of 1. In figure nine this translates to more triangles. However, the triangles can still be separated from the dots with one linear line.

Table 2.

OR classification

x	y	OR
1	1	1
0	1	1
1	0	1
0	0	0

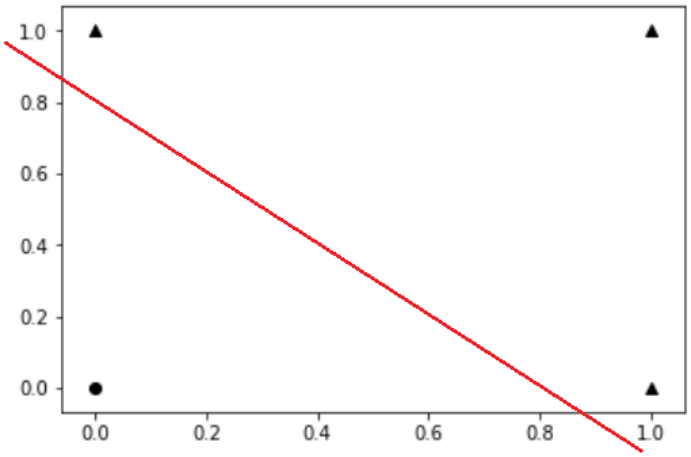

Figure 9: Input space for OR classification. This type of classification is linearly separable as shown by the red line separating the triangles from the dot.

The exclusive/or (EXOR) classification however cannot be separated in a linear manner and can therefore not be classified by a perceptron model. For this classification task the output neuron in the perceptron neuron should take the value of 1 if only x or only y has a value of 1. Figure 10 shows that a single line cannot separate the two classes in the input space.

Table 3.

EXOR classification

x	y	EXOR
1	1	0
0	1	1
1	0	1
0	0	0

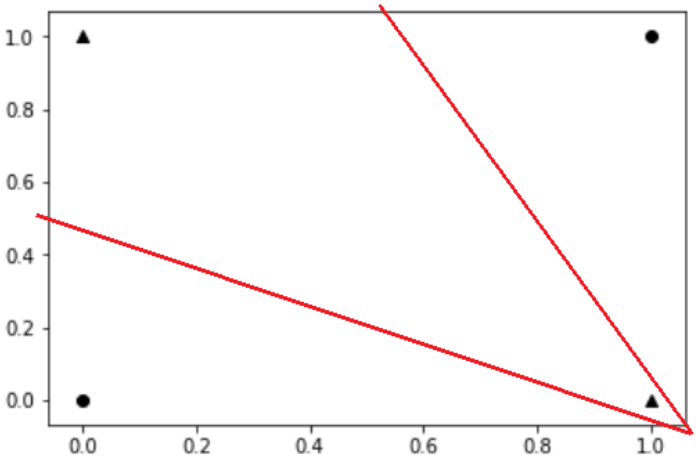

Figure 10: Input space for EXOR classification. Not linearly separable with a single line.

The data used in this study was assumed to not be linear separable and therefore it was expected that this perceptron model could not correctly serve as a control network for the NBA. The perceptron model that was used is shown in figure 11. An input layer with 11 nodes, one for each class in the input data is directly connected to the output layer of 14 nodes. Again, there is one node for each class of the output data. The specific classes in the dataset are discussed in the next section.

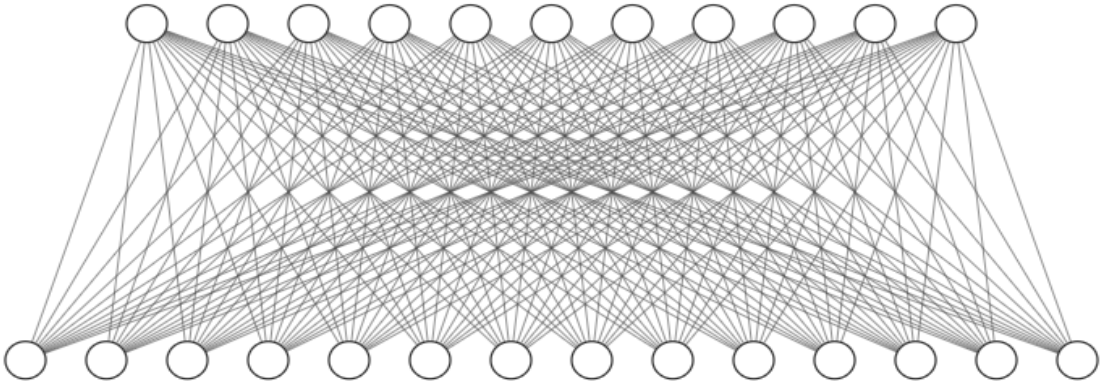

Figure 11: Connection structure for perceptron model with 11 input nodes, 14 output nodes and no hidden layer.

4. Method

In this section the specific in- and output data that were used to model the binding tasks are described. The training and test sentences are given and the way they were encoded to be used by the model is explained. Lastly, the procedure of training the networks and the way in which the networks were evaluated is reported.

4.1 Categories in the in- and output data

The number of nodes in the in-and output layers of the FFN and perceptron model correspond to the number of binding categories that were used. For the input data the different categories and the corresponding nodes in the model are given in table four. The first two nodes in the input layer are dedicated to starting and ending sentences structures. Similar to the study by van der Velde & de Kamps (2006) there are nodes for nouns with a possible relative clause, nouns with a complement clause, verbs and clauses. In addition to these nodes there are five conditional nodes. These conditional nodes cannot be input to the network directly but have to be activated by output first. They then remain active until input either satisfies the condition for the conditional node or new input terminates the activation of the conditional node. An example of how this works is described above in the section on control in the NBA.

Table 4.

Binding categories in the training data

Input layer FFNN

Symbol	*Meaning*	*Node number*
B	Begin S	1
E	End S	2
Nrc	Noun with possible relative clause	3
Ncc	Noun with possible complement clause	4
V	Verb (all verbs have themes)	5
C	Clause (clause word: *that, who)*	6
T	Theme conditional node	7
CC	Complement Clause conditional node	8
CV	Clause-verb conditional node	9
CT	Clause-theme conditional node	10
RC	Relative Clause conditional node	11

For the output layer the first two nodes are dedicated for activating and deactivating structure assemblies. In table five the categories and their corresponding node number are presented. Next there two nodes that encode a noun as either the subject of the sentence as a whole (N-n-S) or as the subject of an embedded clause within the sentence (N-n-C). For binding of a noun as the theme of a verb there is the N-t-V node. The next three nodes are used to bind verbs in the main sentence (V-v-S), verb as the verb of an embedded clause (V-v-C) and clause as a theme of a verb (V-t-C). Binding of a clause as a relative or complement clause to a noun is done with the C-c-N node. Similarly to the input layer the remaining nodes are the conditional ones.

Table 5.

Output symbols (representing output nodes in FFNN)

Output layer FFNN		
Symbol	*Meaning*	*Node number*
S	Activate S assembly	1
-S	Deactivate S assembly	2
N-n-S	Binding N-n-S: noun as subject of sentence	3
N-n-C	Binding N-n-C: noun as subject of clause	4
N-t-V	Binding N-t-V: noun as theme (object) of verb	5
V-v-S	Binding V-v-S: verb as verb of sentence	6
V-v-C	Binding V-v-C: verb as verb of clause	7
V-t-C	Binding V-t-C: Clause word as theme of verb	8
C-c-N	Binding C-c-N: Clause as RC or CC of noun	9
T	T conditional node	10
CC	CC conditional node	11
CV	CV conditional node	12
CT	CT conditional node	13
RC	RC conditional node	14

4.2 Training and testing data

The data that was used to train the network consisted of a set of training sentences that introduced the appropriate input-output relations for the network to be able to process noun-

verb-noun sentences and sentences with an embedded clause. The set of sentences that were used were taken from a paper in which the control is also learned (van der Velde & de Kamps, 2010). Since the neural network cannot process words and sentences directly these were encoded as binary vectors. The input arrays consisted of 11 indices where each index corresponds to one of the binding symbols. This also counts for the output arrays that consist of 14 indices. The index corresponding to the correct binding symbol for a certain word would be encoded as a one and the remaining indices were encoded as zeros. This resulted in arrays as shown in table six. All of the training sentences were stored in a .csv file so it could be read into Python. The complete training and testing data sets can be found in appendix two.

Table 6

Encoding of the data

Binding symbols	**Active nodes**	**Binary encoding**
Nrc (input example)	3	0,0,1,0,0,0,0,0,0,0,0
N-n-S, RC (output example)	3, 14	0,0,1,0,0,0,0,0,0,0,0,0,1

The set of seven test sentences were created so that all the required input-output relations for N-V-N sentences and sentences with an embedded clause were introduced. In total the training set contained three N-V-N sentences and four sentences with an embedded clause. Appendix three contains tables for each of these sentences and the exact input-output relations that were trained and tested. In table seven, the seven sentences that were used to train the model are given. The first three sentences are N-V-N structures that introduce nouns with a possible relative clause (*cat*), verbs and nouns with a complement clause (*fact*). These two different nouns are both introduced in the subject as well as the object position. The remaining sentences introduce the relations needed to bind embedded clauses in the sentence structures.

Table 7.

Set of training sentences that were used for training the network on the input-output relations.

Sentence	Type
Cat sees dog	N-V-N: Nouns with possible relative clauses

Fact worries cat	N-V-N: Noun with possible complement clause in object position.
Cat knows fact	N-V-N: Noun with possible complement clause in subject position.
Cat that chases mouse sees dog	Embedded clause: subject relative
Cat that dog sees chases mouse	Embedded clause: object relative
Cat who fact worries chases mouse	Embedded clause: object relative with Noun with possible complement clause word.
Fact that cat chases mouse worries dog	Embedded clause: Complement clause

In table eight the sentences that were used for evaluating the trained models are shown. These sentences are extensions of the clauses used in the trainings set. In the first test sentence *cat that chases mouse that likes boy sees dog,* there are two relative clauses. The theme of the first relative clause, *cat that chases,* is the relative clause *mouse that likes boy*. In the second test sentence, *cat sees dog that chases mouse,* the relative clause is on the object/theme position. The relative clause *dog that chases mouse* is theme of the verb *sees,* of which *cat is the subject.* The remaining sentences all have multiple embedded clauses in either the subject or object position with the last four sentences combining both complement and relative clauses. The binding that has to occur changes based on whether the complement or relative clause is first in the sentence as well as whether the clause is in the object or subject position in the sentence structure.

Table 8.

Set of test sentences that were used for testing the network on the input-output relations.

Sentence	
Cat that chases mouse that likes boy sees dog	Multiple subject relative clause in subject position
Cat sees dog that chases mouse	Subject relative clause in object position
Cat that dog that boy likes sees chases mouse	Multiple object relative clauses in object position
Cat chases mouse that dog that boy likes sees	Multiple object relative clauses in object position

Fact that cat that boy likes chases mouse worries dog	Complement clause with object relative clause in subject position
Dog knows fact that cat that boy likes chases mouse	Complement clause with object relative clause in object position
Cat who fact that boy likes dog worries chases mouse	Object relative clause with complement clause in subject position
Cat chases mouse who fact that boy likes dog worries	Object relative clause with complement clause in object position

4.3 Training of the network

The materials that were used for the simulations were a laptop with windows and python 3.6 installed. The neural network was created with the Python language using the Keras deep learning library (Chollet, 2015). Programming of the neural network was mainly done on Jupyter Notebook which allows for quick interactive sessions. The training and testing data were stored in a .csv file. The code and the training and testing data can be found in appendix one and two.

The network was trained by feeding the data in a string of arrays to the model. As a training method the method as described by Elman (1993) was used. This entails that the training is performed in several phases. Furthermore, in the study by van der Velde & de Kamps (2010) this method is also applied. Therefore the training was performed in two phases. In the first phase only the simple sentences are trained, in this case these are the noun-verb-noun sentences. The second phase consisted of training with all of the sentences so both the N-V-N sentences and the sentences with an embedded clause. Each of these phases consisted of training with 50.000 epochs. An epoch is an iteration of the model of all of the data points in the trainings dataset. After the training in the first phase the model was saved and the weights were stored so they could be loaded for the training in the second phase.

After the two phase training to replicate the study by van der Velde & de Kamps (2010) effort was given to evaluate the configuration of the model. Mainly the number of nodes in the hidden layer were adjusted to see the effect of this on the performance of the model. It was assumed that the number of hidden nodes was quite high compared with the input and output layer. Having too many nodes in the hidden layer can result in overfitting. Therefore this characteristic of the model was adjusted by decreasing the number of hidden

nodes to evaluate at what point the performance started to be affected. The smallest model that was used consisted of only two layers; an input layer and an output layer. Since there is no hidden layer in this version it is no longer a multilayer FFN, but simply a perceptron model. The connection structure of this model is shown in figure 11. The process of training the models was exactly as described above.

4.4 Evaluation

The neural network will be evaluated by how well it performs on the set of test sentences. Since this simulation is a replication of the study by van der Velde and de Kamps (2010) the network output will be directly compared with the results from that study in which a threshold of .9 was used as minimum activation for desired nodes. Moreover, the other nodes that are not relevant for a given input-output relation should have activation no higher than .1. For every input-output relation in the test set the activation levels were collected and manually checked for adequate activation or spurious activation. In addition to the specific activation levels the accuracy and loss as reported by Keras will be used as determiners of the performance of the network. The accuracy is a measure of the amount of correct predictions the model has made. The loss value is a measure of the distance between the desired output and the output the model computes during the training epochs.

The effect of the hidden layer size was evaluated by running the set of set sentences for each model. The accuracy of these models was compared to judge any performance differences. Additionally, the performance of the model without a hidden layer was inspected in the same manner as the original model. This was done to investigate in more detail whether a simple perceptron model is capable of controlling the binding process for the NBA. The number of correct activations and spurious activations were evaluated for several models for comparison. With these numbers the proportion of correct activations in relation to all the activation of a model was calculated with a simple formula; correct activations / (required activations (170) + number of spurious activations). A perfect model would correctly activate all required nodes and make no spurious activations and therefore have a proportion correct of 170 / (170 + 0) = 1.0. For example a model that made 5 failed activations and 2 spurious activations would give: 170 + 2 = 172 total activations. Of these 172, 7 were incorrect giving (172 – 7) / (170 + 2) = 0.96 proportion correct.

5. Results

5.1 Performance on training data

After the two phases of training the model reported an accuracy of 1.0 and a loss of 0.0015. The maximum accuracy of 1.0 is already reached after approximately 20.000 epochs in both phases. The accuracy of the model during the training is shown in figure 12. After introducing the new sentences with an embedded clause for phase 2 the accuracy initially dropped to .93. This is to be expected when new input-output relations are introduced. With the new data added the accuracy reached 1.0 after approximately 20.000 epochs. The loss kept decreasing until it reached 0.0017 after 50.000 epochs. This accuracy is considered very high and may be an indication of overfitting. Overfitting occurs when the model overoptimizes on the set of training data and learns all the specifics of the training set that cannot be generalized to other new data (Chollet, 2018). This would lead to less predictive power for the test data. The input-output relations in the training set are learned to perfection and are therefore classified correctly when fed to the model after training. For every output the activation for each node ranges between 0 and 1. The correct outputs are produced by the model for the training set with activation levels above 0.9 and there are no spurious activations (< 0.1). This is similar to the results by van der Velde & de Kamps (2010). The next step to determine the performance of the model was to test it with the set of test sentences. These sentences are based on the training set with different combinations and extensions of the clauses.

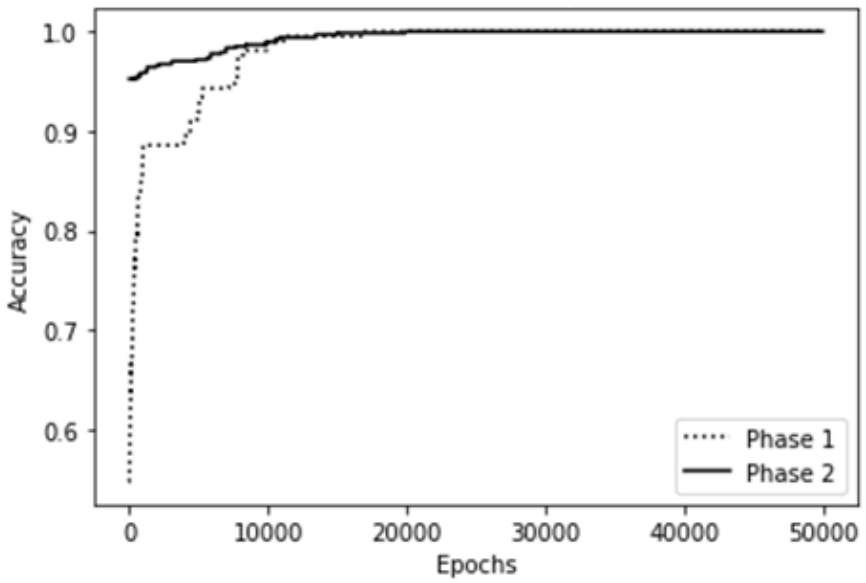

Figure 12. Model accuracy for two phase training with 50.000 epochs.

5.2 Performance on test data

Evaluation of the fully trained model in Keras gives an accuracy of .99 and a loss of 0.066 for the test data. These results are comparable to the results on the training data and therefore it seems the model has not overfit. The output activation for each word in the set of test sentences was evaluated to see the performance in detail.

The first sentence in the test set is an extension of a sentence with an embedded clause in which the theme in a relative clause also has a relative clause. The trained model produced the correct output for this sentence. Every required output node had an activation higher than 0.9 and the other nodes were all below 0.1 meaning that there was no spurious activation. The exact activation levels for each input are given in table nine.

Table 9

Activation levels (AL) as given by the model with 12 hidden nodes for the test sentence: Cat that chases mouse that likes boy sees dog.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Cat	3 (.98), 14(.99)	
That	9 (.99), 12 (.99), 14 (.99)	
Chases	7 (.98), 10 (.98)	
Mouse	5 (.99), 14 (1)	
That	9 (.99), 12 (.99), 14 (.99)	
Likes	7 (.98), 10 (.98)	
Boy	5 (.99), 14 (1)	
Sees	6 (.99), 10 (.99)	
Dog	5 (.99), 14 (1)	

The second sentence in the test set, *Cat sees dog that chases mouse*, has a subject relative clause in the object position. The model was only trained with subject relative clauses in the subject position. However, the trained model correctly activated the required nodes with levels above 0.9 and again no spurious activations as shown in table 10 .

Table 10

Activation levels (AL) as given by the model with 12 hidden nodes for the test sentence: Cat sees dog that chases mouse.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**

Cat	3 (.98), 14 (.99)
Sees	6 (.99), 10 (.99)
Dog	5 (.99), 14 (1)
That	9 (.99), 12 (.99), 14 (.99)
Chases	7 (.98), 10 (.98)
Mouse	5 (.99), 14 (1)

The third sentence in the test set is one with a multiple object-relative clause and is difficult for humans to understand (van der Velde & de Kamps, 2010). However, the model was able to correctly activate the required binding categories for all the input-output relations. There were no spurious activation and all the required activation were above 0.9 as shown in table 11.

Table 11

Activation levels (AL) as given by the model with 12 hidden nodes for the test sentence: Cat that dog that boy likes sees chases mouse.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Cat	3 (.98), 14(.99)	
That	9 (.99), 12 (.99), 14 (.99)	
Dog	4 (.99), 13 (.93), 14 (1)	
That	9 (.99), 12 (.99), 14 (.99)	
Boy	4 (.99), 13 (.93), 14 (1)	
Likes	7 (.99), 8 (.98)	
Sees	7 (.99), 8 (.98)	
Chases	6 (.99), 10 (.99)	
Mouse	5 (.99), 14 (1.0)	

In table 12 the activation levels for the test sentence, *Cat chases mouse that dog that boy likes sees*, are given. Again all the required nodes are activated with levels above .9 and there are no spurious activations.

Table 12

Activation levels (AL) as given by the model with 12 hidden nodes for the test sentence: Cat chases mouse that dog that boy likes sees.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Cat	3 (.98), 14(.99)	
Chases	6 (.99), 10 (.99)	
Mouse	5 (.99), 14 (1)	
That	9 (.99), 12 (.99), 14 (.99)	
Dog	4 (.99), 13 (.93), 14 (1)	
That	9 (.99), 12 (.99), 14 (.99)	
Boy	4 (.99), 13 (.93), 14 (1)	
Likes	7 (.99), 8 (.98)	
Sees	7 (.99), 8 (.98)	

For the test sentence: *Fact that cat that boy likes chases mouse worries dog* all input-output relations were processed correctly by the trained model. All with activations above 0.9 and without spurious activation as shown in table 13.

Table 13

Activation levels (AL) as given by the model with 12 hidden nodes for the test sentence: Fact that cat that boy likes chases mouse worries dog.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Fact	3 (.99), 11 (.99)	
That	9 (.98), 11 (.96), 12 (.98)	
Cat	4 (.93), 14 (.99)	
That	9 (.99), 12 (.99), 14 (.99)	
Boy	4 (.99), 13 (.93), 14 (1)	
Likes	7 (.99), 8 (.98)	
Chases	7 (.98), 10 (.98)	
Mouse	5 (.99), 14 (1)	
Worries	6 (.99), 10 (.99)	
Dog	5 (.99), 14 (1)	

In table 14 the activation levels for the test sentence *dog knows fact that cat that boy likes chases mouse* are given. For every individual input in this sentence structure the correct

binding categories are activated by the trained model. All of the activation are higher than .9 and therefore sufficient to create the appropriate bindings. No other nodes were activated with a level higher than .1 meaning there is no spurious activation for this sentence structure.

Table 14

Activation levels (AL) as given by the model with 12 hidden nodes for the test sentence: Dog knows fact that cat that boy likes chases mouse.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Dog	3 (.98), 14 (.99)	
Knows	6 (.99), 10 (.99)	
Fact	5 (.98), 11 (.99)	
That	9 (.98), 11 (.96), 12 (.98)	
Cat	4 (.93), 14 (.99)	
That	9 (.99), 12 (.99), 14 (.99)	
Boy	4 (.99), 13 (.93), 14 (1)	
Likes	7 (.99), 8 (.98)	
Chases	7 (.98), 10 (.98)	
Mouse	5 (.99), 14 (1)	

In the remaining sentences in the test set there are some outputs that the model did not activate correctly. For the sentence, *Cat who fact that boy likes dog worries chases mouse,* an error was made by the model. Firstly, for the verb *likes* the wrong binding categories were activated. The verb is correctly bound as the verb of a clause (V-v-C). However, the node for theme (T, node 10) is not correctly activated and instead the node for clause as theme of a verb (V-t-C, node 8) is activated. This In table 15 the activation levels for these nodes is shown.

Table 15

Activation levels (AL) as given by the model with 12 hidden nodes for the test sentence: Cat who fact that boy likes dog worries chases mouse.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Cat	3 (.98), 14 (.99)	
Who	9 (.99), 12 (.99), 14 (.99)	

Fact	4 (.99), 11 (.99), 13 (.98)	
That	9 (.98), 11 (.96), 12 (.98)	
Boy	4 (.99), 13 (.91), 14 (1)	
Likes	7 (.99), 10 (.01)*	8 (.98)
Dog	5 (.99), 14 (1)	
Worries	7 (.99), 8 (.98)	
Chases	6 (.99), 10 (.99)	
Mouse	5 (.99), 14 (1)	

Note: *AL below threshold of .9.

In the last test sentence of the test set for the verb *likes* the theme node (T) is not correctly activated. Instead the node for binding a clause word as theme of a verb (V-t-C) is activated. The activation levels for this sentence are given in table 16. The error made by the model is the same error as the one that was made in the test sentence shown in table 10. This shows that the model was not able to learn to activate the desired nodes for these input-output relations. This result will be discussed more extensively in the discussion section.

Table 16

Activation levels (AL) as given by the model with 12 hidden nodes for the test sentence: Cat chases mouse who fact that boy likes dog worries.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Cat	3 (.98), 14 (.99)	
Chases	6 (.99), 10 (.99)	
Mouse	5 (.99), 14 (1)	
Who	9 (.99), 12 (.99), 14 (.99)	
Fact	4 (.99), 11 (.99), 13 (.98)	
That	9 (.98), 11 (.96), 12 (.98)	
Boy	4 (.93), 14 (.99)	
Likes	7 (.99), 10 (.01)*	8 (.98)
Dog	5 (.99), 14 (1)	
Worries	7 (.99), 8 (.98)	

Note: *AL below threshold of .9.

With the exception of the last two sentences the results of van der Velde and de Kamps (2010) could be replicated. The first six sentences of the test set are processed without any problems by the trained model. All of the bindings in these sentences proceed correctly, under the assumption that the conditional nodes are activated in the NBA.

5.3 Effect of number of nodes in the hidden layer

After these first simulations the configuration of the model was adjusted to study the effect on the performance. Mainly, the number of hidden nodes was lowered from the initial twelve to no nodes/hidden layer at all. All other configurations and training steps were kept the same as the replication with the original model described above. In figure 13 the effect on the model accuracy is shown. For each number of nodes in the hidden layer the performance for processing the set of training and test sentences was evaluated. The change in accuracy across the varying amount of hidden neurons is slightly erratic. With as little as seven hidden nodes the accuracy of the model for the predicting the test data remains near perfect. It is only with less than six nodes that the accuracy starts to drop but not in a significant amount. The accuracy stays well above a level of .96 until a model with only three nodes in the hidden layer. The model with a hidden layer consisting of one or two hidden nodes has the lowest accuracy. Even lower than a model that has no hidden layer at all. In fact, the model in which the hidden layer was completely omitted showed a performance similar to the original model with a hidden layer consisting of 12 nodes.

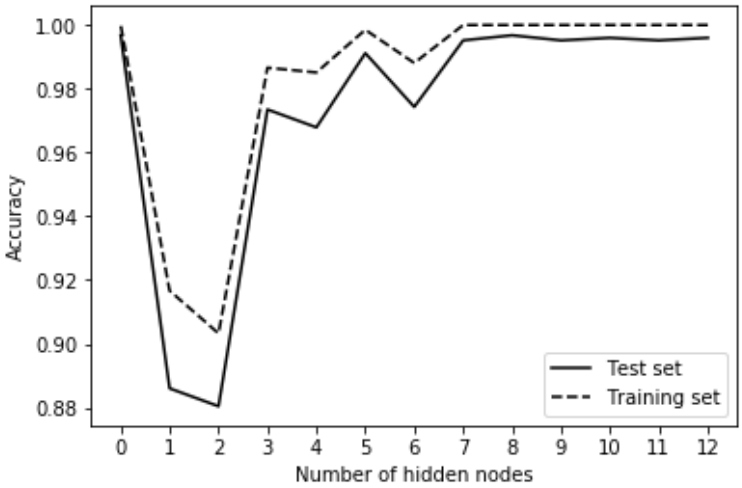

Figure 13. Accuracy for each number of nodes in the hidden layer.

The loss, shown in figure 14, shows a similar pattern across the varying hidden layer sizes. Loss is the highest for the models with one and two hidden nodes. For hidden layer sizes between seven and twelve the loss remains relatively low. In parallel with the accuracy, the loss for the model without a hidden layer is similar to that of the original model with twelve hidden nodes.

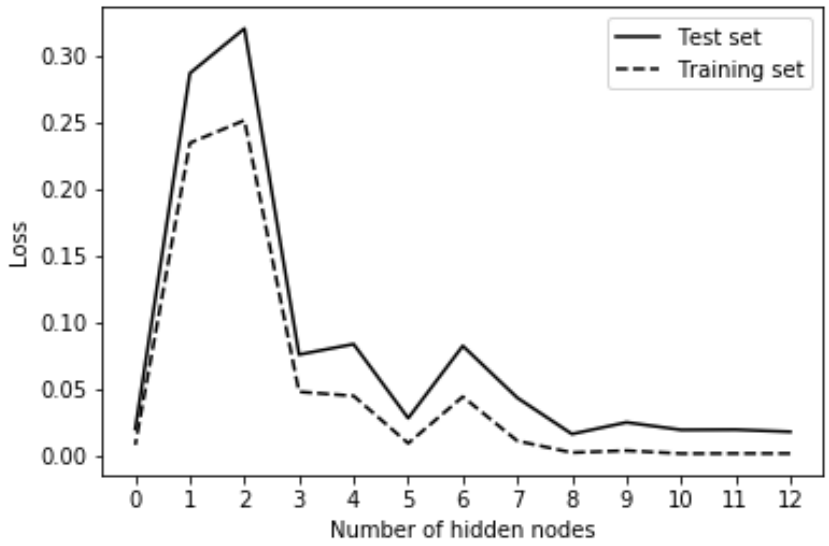

Figure 14. Loss shown per model with different number of nodes in the hidden layer.

To investigate the effect of the hidden layer even further all of the test sentences where processed with the model with zero hidden nodes. The accuracy for this model is just as high as a model with as many as 12 hidden nodes. This is quite remarkable and therefore all input-output relations in the test set are examined below. The connection structure of the model without a hidden layer is shown in figure 12 in the architectures section. This shows that each input node is directly connected to all nodes in the output layer. The input data thus, does not pass through a hidden layer first.

In table 17 the first sentence is presented; *Cat that chases mouse that likes boy sees dog.* All the required nodes are activated with sufficient levels of activation (> .9). However, in two cases there is spurious activation. For *chases* and *likes* the eighth node is spuriously activated with a level of .1. This node should only be activated when the word is a clause word as a theme of a verb. This sentence did not present these difficulties for the original model with a hidden layer consisting of 12 nodes.

Table 17

Activation levels (AL) as given by the model with 0 hidden nodes for the test sentence: Cat that chases mouse that likes boy sees dog.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Cat	3 (.95), 14(.99)	
That	9 (.98), 12 (.99), 14 (.97)	
Chases	7 (.93), 10 (.91)	8 (.10)
Mouse	5 (.99), 14 (.99)	
That	9 (.98), 12 (.98), 14 (.96)	
Likes	7 (.93), 10 (.91)	8 (.10)
Boy	5 (.99), 14 (.99)	
Sees	6 (.98), 10 (.99)	
Dog	5 (.99), 14 (.99)	

In table 18 the test sentence *cat sees dog that chases mouse* is presented. Again, for a verb, *chases*, node eight is spuriously activated. The rest of the binding categories are activated correctly.

Table 18

Activation levels (AL) as given by the model with 0 hidden nodes for the test sentence: Cat sees dog that chases mouse.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Cat	3 (.95), 14(.99)	
Sees	6 (.98), 10 (.99)	
Dog	5 (.99), 14 (.99)	
That	9 (.98), 12 (.98), 14 (.96)	
Chases	7 (.93), 10 (.91)	8 (0.10)
Mouse	5 (.99), 14 (.99)	

In table 19 the activation levels are presented for the test sentence: *cat that dog that boy likes sees chases mouse.* All the required nodes are activated without any spurious activations. However, there are several nodes with activation levels below .9. The nouns *dog* and *boy* have to be bound as the subject of a clause (N-n-C). However, in both cases the activation for this node is only .87. The same counts for node 13, which binds a noun as the

clause theme. For *dog* and *boy* the activation level is .8. additionally, For the two verbs, *likes* and *sees* in this test sentence there is an insufficient activation level for node 8.

Table 19

Activation levels (AL) as given by the model with 0 hidden nodes for the test sentence: Cat that dog that boy likes sees chases mouse.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Cat	3 (.95), 14(.99)	
That	9 (.98), 12 (.98), 14 (.96)	
Dog	4 (.87)*, 13 (.80)*, 14 (.99)	
That	9 (.98), 12 (.98), 14 (.96)	
Boy	4 (.87)*, 13 (.80)*, 14 (.99)	
Likes	7 (.99), 8 (.89)*	
Sees	7 (.99), 8 (.89)*	
Chases	6 (.98), 10 (.99)	
Mouse	5 (.99), 14 (.99)	

Note: * AL below the threshold of .9.

As shown in table 20, for the next test sentence the model with no hidden layer makes the same mistakes as earlier. Namely, the activation levels for nodes 4 and 13 for two noun inputs are insufficient. And node 8 is not activated adequately for a verb input. The model with a hidden layer of 12 nodes processed this test sentence without any difficulties.

Table 20

Activation levels (AL) as given by the model with 0 hidden nodes for the test sentence: Cat chases mouse that dog that boy likes sees.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Cat	3 (.95), 14(.99)	
Chases	6 (.98), 10 (.99)	
Mouse	5 (.99), 14 (.99)	
That	9 (.98), 12 (.98), 14 (.96)	
Dog	4 (.87)*, 13 (.80)*, 14 (.99)	
That	9 (.98), 12 (.98), 14 (.96)	

Boy	4 (.87)*, 13 (.80)*, 14 (.99)
Likes	7 (.99), 8 (.89)*
Sees	7 (.99), 8 (.89)*

Note: *AL below .9.

Activation levels for *fact that cat that boy likes chases mouse worries dog* are shown in table 21. The model without hidden layer has made several mistakes in processing this sentence. The first occurs at the clause word *that.* Here, node 11 has an insufficient activation level of .88. The next word in the sentence, *cat,* is also not processed correctly. Firstly, node 4 has a low activation (.63). Secondly, there is spurious activation for node 3 (AL .15). For the word *boy* the same activation levels before are reported. Node 4 and 13 have insufficient activation levels. Lastly, there is a slightly too low activation for node 8 at the verb *likes* and there is spurious activation for the verb *chases.*

Table 21

Activation levels (AL) as given by the model with 0 hidden nodes for the test sentence: Cat chases mouse that dog that boy likes sees.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Fact	3 (.95), 11 (.99)	
That	9 (.98), 11 (.88)*, 12 (.97)	
Cat	4 (.63)*, 14 (.97)	3 (.15)
That	9 (.98), 12 (.98), 14 (.96)	
Boy	4 (.87)*, 13 (.80)*, 14 (.99)	
Likes	7 (.99), 8 (.89)*	
Chases	7 (.93), 10 (.91)	8 (.10)
Mouse	5 (.99), 14 (.99)	
Worries	6 (.98), 10 (.99)	
Dog	5 (.99), 14 (.99)	

Note: *AL below .9.

Activation levels for the test sentence *Dog knows fact that cat that boy likes chases mouse* are given in table 22. There are several processing errors for the input-output relations in this sentence structure. The first occurs at the clause word *that*. This input requires the complement clause node (node 11) to become activated. This activation would signal the next

word in the sentence could be part of a complement clause. However, the activation is insufficient at a level of .88. the next word *cat* should then be bound as a noun as the subject of a clause (N-n-C, node 4). This binding fails due to an activation level of .63, which is too low. Additionally, the node for binding a noun as the subject of a sentence structure is spuriously activated (AL .15 for node 3). Another two insufficient activations occur at the noun *boy*. Node three and 13 are both too low which means *boy* is not correctly bound as a noun as subject of a clause (N-n-C). Additionally, the clause theme conditional node is not activated. This would result in a failure to bind the embedded clause *cat that boy likes* in which *boy* is the theme of *likes.* This is also reflected in the fact that at *likes* the node for binding a word as the theme of a verb (V-t-C) is not sufficiently activated (AL for node 8 = .89). Lastly, in this sentence structure node 8 is spuriously activated for *chases.*

Table 22

Activation levels (AL) as given by the model with 0 hidden nodes for the test sentence: Dog knows fact that cat that boy likes chases mouse.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Dog	3 (.95), 14 (.99)	
Knows	6 (.98), 10 (.99)	
Fact	5 (.98), 11 (.99)	
That	9 (.98), 11 (.88)*, 12 (.97)	
Cat	4 (.63)*, 14 (.96)	3 (.15)
That	9 (.98), 12 (.98), 14 (.96)	
Boy	4 (.87)*, 13 (.80)*, 14 (.99)	
8Likes	7 (.99), 8 (.89)*	
Chases	7 (.93), 10 (.91)	8 (.10)
Mouse	5 (.99), 14 (.99)	

Note: *AL below .9.

In table 23 the activation output for *cat who fact that boy likes dog worries chases* is shown. This is also the first of the two sentences that was not processed correctly by the original model with 12 hidden nodes. The activation levels follow a similar pattern to those of the sentences presented above. There is a combination of activation levels that do not reach the required level of .9 and spurious activations. The first insufficient activation occurs at *fact* should be bound as noun of a complement clause (N-n-C, node 4). However, this binding fails

with a too low activation level of .88. At *fact* a clause theme should also be anticipated with the conditional node CT (node 13). Again, this binding fails.

Table 23

Activation levels (AL) as given by the model with 0 hidden nodes for the test sentence: Cat who fact that boy likes dog worries chases mouse.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Cat	3 (.95), 14 (.99)	
Who	9 (.98), 12 (.98), 14 (.96)	
Fact	4 (.88)*, 11 (.98), 13 (.87)*	
That	9 (.98), 11 (.88)*, 12 (.97)	
Boy	4 (.99), 13 (.90), 14 (.96)	5 (.22), 12 (.10)
Likes	7 (.99), 10 (.07)*	8 (.89)
Dog	5 (.99), 14 (.99)	
Worries	7 (.99), 8 (.89)*	
Chases	6 (.98), 10 (.99)	
Mouse	5 (.99), 14 (.99)	

Note: *AL below .9.

The activation levels as given by the model with zero hidden nodes for the test sentence *cat chases mouse who fact that boy likes dog worries* are given in table 24. At the input word *fact* the first error occurs. The activation levels for node four and 13 are too low. Another node that is not sufficiently activated is node 11 for the input word *that.* For *boy* there is both too little activation and spurious activation. The same counts for the verb *likes*. Here the same error occurs as with the model with 12 hidden nodes.

Table 24

Activation levels (AL) as given by the model with 0 hidden nodes for the test sentence: Cat chases mouse who fact that boy likes dog worries.

Word	**Required nodes (AL)**	**Spurious nodes (AL)**
Cat	3 (.95), 14 (.99)	
Chases	6 (.98), 10 (.99)	
Mouse	5 (.99), 14 (.99)	

Who	9 (.98), 12 (.98), 14 (.97)	
Fact	4 (.88)*, 11 (.98), 13 (.87)*	
That	9 (.98), 11 (.88)*, 12 (.97)	
Boy	4 (.63)*, 14 (.97)	3 (.15)
Likes	7 (.99), 10 (.07)*	8 (.89)
Dog	5 (.99), 14 (.99)	
Worries	7 (.99), 8 (.89)*	

Note: *AL below .9.

After reviewing the activation levels for every input-output relation in the test set for both the model with 12 hidden nodes and the model without a hidden layer, it becomes clear that the latter is off on way more activations than the former. Where the original model processed 6 of the 8 test sentences flawlessly, the model without a hidden layer made mistakes in every sentence. In table 25 some descriptive statistics regarding the models are presented. The performance of the models is indicated with the number of unsuccessful bindings and spurious bindings. The model with a hidden layer size of 12 nodes had too low activation levels for two nodes. On the other hand, the model without a hidden layer made 33 inadequate activations. However, the model without a hidden layer is not the worst performing model. As can be seen in table 25 and figure 15, the models with ranging from one to four and the models with six or seven nodes all perform worse. All of these models have a higher number of failed bindings, more spurious bindings and therefore a lower proportion correct out of the total activations made. That being said, this analyse paints a different picture than the accuracy as reported by Keras. The model without a hidden layer is not capable of flawlessly controlling the binding for the NBA according to these criteria.

Table 25

Overview of the models based on number of failed activations, spurious activations and proportion correct activations.

Model	**Number of failed bindings; AL < .9**	**Number of spurious bindings; AL > .1**	**Proportion correct activations out of total activations**
12 hidden nodes	2	2	0.98
11 hidden nodes	2	9	0.94
10 hidden nodes	2	3	0.97

9 hidden nodes	14	7	0.88
8 hidden nodes	3	4	0.96
7 hidden nodes	44	13	0.69
6 hidden nodes	76	86	0.37
5 hidden nodes	19	17	0.81
4 hidden nodes	64	86	0.41
3 hidden nodes	76	62	0.41
2 hidden nodes	170	385	0
1 hidden node	129	417	0.07
No hidden layer	33	12	0.75

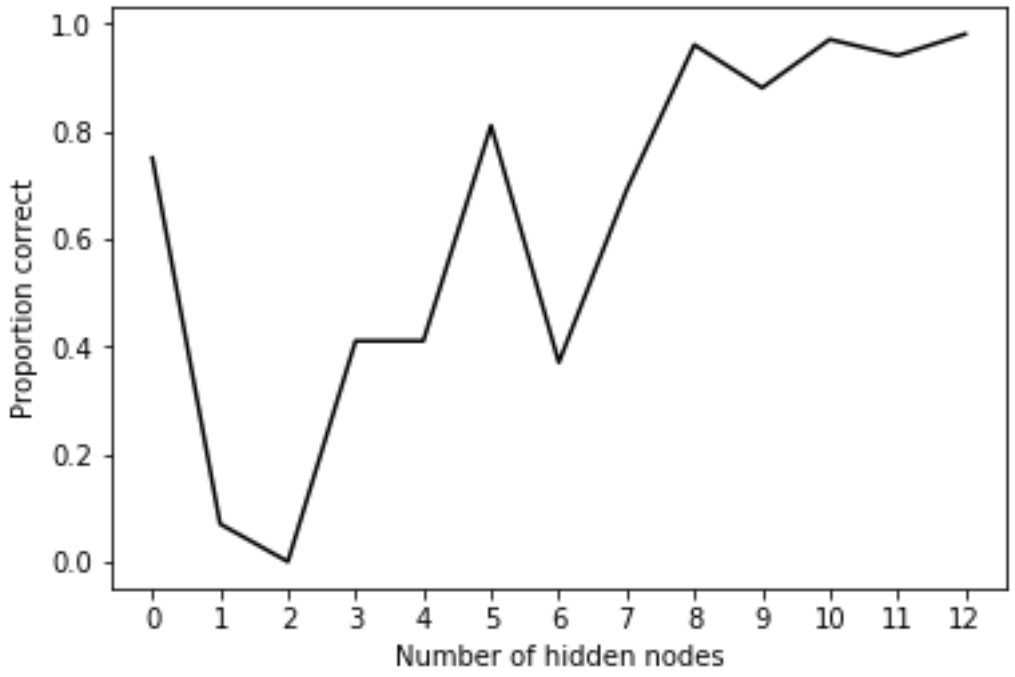

Figure 15. Proportion of correct activations in relation to total activations made for each number of hidden nodes.

6. Discussion

6.1 Conclusions

In this thesis a control network was tested that aims to operate according to three principals of human cognition. Namely, dynamic, grounding and productivity. The aim was to replicate and expand the study by van der Velde and de Kamps (2010) to see if and under which conditions a simple feedforward neural network can learn input and output relations to control the binding in the neural blackboard architecture of sentence structure (van der Velde & de Kamps, 2006).

The results of van der Velde and de Kamps (2010) could be replicated almost completely with the network presented in this thesis. The first six sentences in the test set were processed perfectly by the model with all the required nodes activated at levels of at least .9 and no spurious activation. In the last two sentences of the test set the model incorrectly activated output nodes. In both cases this error occurred when the verb *likes* was encountered. The fact that the word *likes* does not get classified correctly is interesting. This is in fact a case of an unproblematic ambiguity. An unproblematic ambiguity arises when a local ambiguity in a sentence causes no difficulty (Lewis, 1993). The phenomenon can be illustrated with a simple example. Consider the sentence *Shelley knows Tom.* In this simple noun-verb-noun sentence *Tom* is the theme of the verb *knows*. Now consider the sentence *Shelley knows Tom likes skateboarding*. The parsing of this sentence proceeds in the same way as the first sentence, until the verb *likes* is encountered. Then the whole clause *Tom likes skateboarding* becomes the theme of the verb *knows.* For the test sentence *Cat who fact that boy likes dog worries chases mouse* such an ambiguity occurs with the verb *likes.* As soon as the verb *likes* is encountered there are two possibilities. It can be bound to *boy* with *boy* being the agent of *likes* in the clause *boy likes dog*. Alternatively *likes* can be bound as the verb of the clause *Cat who boy likes.* In this case *cat* is the *theme* of the verb *likes.* These two different interpretations are both linguistically correct. However, they differ in the amount of difficulty they would present to the human reader during parsing. This also happens for the sentence *cat chases mouse who fact that boy likes dog worries.* Similarly the ambiguity arises at the verb *likes.* Again, *likes* can be bound to boy with boy being the object of *likes* in the clause *boy likes dog.* The second interpretation would be that *likes* is the verb of the clause *who fact that boy likes* with *mouse* being the theme of *likes.*

Since the trained model is a feedforward network it cannot correct via backtracking when such an ambiguity is encountered. It simply processes a sentence word for word. This means that the model sticks with the first interpretation which, in the case of an ambiguity,

could result in an erroneous binding. Other sentences that contain such an ambiguity would therefore also be processed incorrectly by the trained model. The wrong binding symbols are then activated and given to the NBA. In the trained FFN there is not really a way to deal with such ambiguities. When the data is pre-processed a choice for one of the two possibilities is made based on which alternative is more logical. The NBA can resolve such ambiguities circumventing the problem of the FFN concerning ambiguities. This is done by having a competition between the two conflicting binding activations. How this process works is illustrated in the paper by van der Velde and de Kamps (2015). In this paper the unproblematic ambiguity *Bill knows John likes fish* is simulated in the NBA. Firstly, *Bill knows* activates a sentence assembly and binds *Bill* as a noun and subject of the sentence. *Knows* is represented as the verb of the main of the sentence. When *John* is encountered the third binding takes place. *John* is bound as the theme of *knows.* When the rest of the sentence is encountered this third binding will become deactivated. This happens when *likes* is processed and a complement clause is introduced. At this point a conflict arises because *likes* is now bound as the verb of the clause *John likes fish.* This clause should now be bound as a complement clause to the verb *knows*. In the simulation this is exactly what happens showing that the NBA can resolve such ambiguities (van der Velde & de Kamps, 2015).

The effect of the hidden layer size was tested by gradually lowering the amount of hidden neurons and eventually removing the hidden layer completely. This showed that overall a model with less neurons in the hidden layer reaches a lower accuracy after the training. However, when the hidden layer was removed completely the performance of the model seemed similar to the performance of the original model with twelve hidden nodes based on the reported accuracy by Keras. Using more than one hidden layer seems unnecessary for most problems. Even for highly complex and non-linear problems (Shafi, Ahmad, Shah & Kashif, 2006; Panchal & Panchal, 2014). Having more than one hidden layer only increases the complexity of a model and can decrease the performance of the model. The number of hidden neurons can also have an effect on the performance of a model. Selecting too few nodes can result in underfitting since there are two little neurons to capture the relations in a dataset. On the other hand, having too many neurons can results in overfitting. In this case the model will start to over adapt to the training dataset and learn relations that are very specific for the training data and will not generalize to new data (Panchal & Panchal, 2014; Chollet, 2018). Additionally, too many hidden neurons can increase the required training time and therefore make training more resource intensive (Panchal & Panchal, 2014). Regarding the choice for the number of hidden neurons some rules of thumb have been

suggested. The number of hidden neurons should be between the size of the input and the size of the output layer and the number of hidden neurons should be no more than twice the size of the input layer (Panchal & Panchal, 2014).

Related to the number of hidden layers/neurons in a FFN is the number of trainable parameters. This is a direct result of the number of layers and nodes in the network architecture as well as the dimensionality of the data. The number of learnable parameters is also referred to as the capacity of the model (Chollet, 2018). More parameters means more capacity to memorize the relations between input and output data. This also means that having too many learnable parameters could result in the model simply memorizing the exact relations without it being able to generalize. For instance, a model with 48 times 14 (672) trainable parameters can simply use one binary parameter for all 48 samples which can have 14 possible classes. Conversely, too few parameters could mean that the model has too little capacity to actually learn anything. However, having less parameters and therefore limited memory capacity could be an advantage. The model will then have to learn compressed representations that have predictive power. These are exactly the type of representations the model should learn to be able to generalize (Chollet, 2018).

The information above could be used as an explanation for the fact that the model architecture without a hidden layer actually outperformed models with hidden layer sizes ranging between one and seven hidden neurons. The model without a hidden layer actually has more trainable parameters (378) than a model with as many as seven hidden nodes (252). This configuration therefore has more capacity to be able to learn all the relations in the data. The same is true for the models with eight or more hidden nodes. The optimal model therefore seems to be the model with the smallest hidden layer size and trainable parameters in relation to the accuracy. This configuration uses the least amount of trainable parameters out of the models that reach the highest accuracy on the testing and training sets. A configuration of this kind in desirable since it reduces complexity of the model and the resources needed to train the model (Radford et al., 2018).

In addition to the network architecture the dimensionality of the data itself should be considered. A difference between a neural network with or without a hidden layer is the capability to handle data that is linear separable. A perceptron can only handle data that is linearly separable. To deal with more complex non-linear problems a model with at least one hidden layer should be used. Since the data that was used for this control network is assumed to be non-linear it was expected that a perceptron model could not be used. The first accuracy reported by Keras seemed to point to a well working perceptron model for controlling the

binding. However, upon closer expectation it became clear that many bindings failed with the perceptron model. Still, when the requirements from the paper by van der Velde and de Kamps (2010) were applied the perceptron model correctly activated 75% of the nodes needed for binding in the test set. This could suggest that indeed a large portion of the data that was used is linear separable. Moreover, it means that a choice has to be made regarding the criteria to use when implementing the control network. If the standard Keras accuracy metric is used the model without a hidden layer seems to be sufficient for the control task. This is because Keras calculates this accuracy with a threshold of .5. In addition to this all of the zeros in the input vectors are also checked against this threshold. The threshold that was set for this study was a minimum of .9 for activation to be correct and for spurious nodes the activation could not exceed .1. This means that activity for non-required nodes between .1 and .49 is seen as accurate by Keras but is incorrect when the .9 threshold is applied. Moreover, activity for required nodes from .5 and upwards was sees as correct by Keras. This is likely what caused the high accuracy metrics even for the models that made as many as 33 binding errors when checked manually against the .9 threshold. The differences between the models are still accurate; the model without a hidden layer still outperforms the models rating from 1 to 7 hidden nodes. When a custom accuracy threshold of .9 is used the optimal models seem to be those with 11, 10 or 8 hidden nodes since those models reach high accuracies with less capacity than the original model with 12 hidden nodes.

Based on the trend of the replication model accuracy it may be suggested that the number of epochs during training in the original paper was too high. Namely, the accuracy did not improve any further after approximately 20.000 epochs. Still, the model was trained for an additional 30.000 epochs. This can be problematic as the possibility for overfitting occurs when the model continues to train after the accuracy stops improving. The loss can be monitored to see if the model is still improving. Since the loss kept decreasing during the 50.000 epochs it was concluded that the model was not overfitting. However, training the model for longer than 20.000 epochs is not necessary either and training can be stopped at this point.

6.2 Limitations

The FFNs that were used were successful on the test of combinatorial productivity. The model could correctly process most of the input-output relations in the test set. However, this set is still quite limited and therefore the control network presented in this thesis is not yet a network that is on the same scale as human combinatorial productivity. It does confirm the

finding of van der Velde & de Kamps (2006) that a FFN can be used to simulate the learning of grounded combinatorial structures.

A first methodological limitation of this study regards the method for evaluating the model. During this evaluation there was no use of validation data. Normally, a portion of the training dataset is set aside for validation. During training the model then also explores the ability of the model to classify the data in this validation set. Since these are samples that were not trained on this functions as a way to check if the model is able to generalize to new data. Overfitting can then be easily determined by plotting the accuracy for both the training and validation data. If the performance of the model keeps increasing for the trainings data but stalls or decreases for the validation set the model is overfitting. In the current study this procedure was not possible due to the limited amount of training data available. Only the set of sentences taken from the paper of van der Velde and de Kamps (2010) were used. This is a relatively small dataset for training to begin with. Moreover, the sentences in the training dataset were constructed in such a way that all the required input-output relations were in the set once. Therefore, setting aside a portion of the data for validation seemed unfeasible since this would have removed crucial data characteristics for training. In future studies this could be accounted for by creating a larger dataset of more sentences. This would allow the possibility of using validation steps during the training and will lead to better evaluations of the performance of the model. However, creating more instances of an input-output relation does not make sense because it is not the specific word type information the model learns on. Instead, the abstract classes, such as nouns and verb, are learned. For instance, another noun-verb-noun sentence can be created to use for validation. However, creating sentences with different nouns and verbs does not change the relations between them in any way and is therefore not really different. In fact, the entire principle of combinatorial productivity relies on learning a sentence structure in which the entities can be replaced with any arbitrary noun or verb but still be understood.

A second limitation is related to the assumption that was made in regards to activity in the NBA. The conditional nodes that become active as output before they can be used as input in the control network. In the NBA these conditional nodes stay active until the nodes are either activated in conjunction with new input, such as an incoming relative clause, or until a new input cancels the activation. This means that the success of the FFN in controlling the binding is also due to the assumed feedback of the NBA. This part of the simulation was based on the assumption that this process would work this way. However, it was not actually tested with the NBA.

6.3 Recommendations and future directions

Based on the results and discussion some recommendations for future studies can be made. The first recommendations is to add more sentence structures in the training. The model in this study was only trained on two sentence structures. Namely, noun-verb-noun sentences and sentences with an embedded clause. Unproblematic ambiguities were already briefly discussed by could be explored further. Additionally, garden-path effects could be examined. A garden-path sentence concerns an ambiguity that cannot be resolved without re-parsing of the sentence (Lewis, 1993). Based on the results in this study it is expected that garden path sentences will present difficulties to the feedforward network similarly to the unproblematic ambiguity that was encountered.

A second recommendation is to extend the model by training it on more word types. Examples of these word types are adjectives, adverbs and determinative words. This would actually require a different network with more nodes. New word types will introduce new types of relations between them and therefore more binding categories will have to be learned by the model. This sort of extension of the model requires it to be completely retrained on a new dataset. Furthermore, the optimal number of nodes in the hidden layer would have to determined again.

7. Literature

Chollet, F. (2018). *Deep learning with Python.* Shelter Island, NY: Manning publications.

Elman, J. L. (1991) Distributed representations, simple recurrent networks, and grammatical structure.

Elman, J.L. (1993). Learning and development in neural networks: the importance of starting small. *Cognition, 48,* 71-99.

Fodor, J. A., & Pylyshyn, Z. W. (1988) Connectionism and cognitive architecture: A critical analysis. *Cognition 28,* 3–71. Available at: http://citeseer.ist.psu.edu/446398.html.

Hebb, D. O. (1949) The organization of behavior: A neuropsychological theory. Wiley.

Jackendoff, R. (2002). *Foundations of Language*. Oxford: Oxford University Press.

Lambon-Ralph, M. A., Jefferies, E., Patterson, K., & Timothy T. Rogers, T. T. (2017). The neural and computational bases of semantic cognition. *Nature Reviews Neuroscience 18,* 42–55.

Lewis, R. L. (1993). *An Architecturally-based Theory of Human Sentence Comprehension.* Thesis Carnegie Mellon University, Pittsburgh, PA.

Panchal, F.S., & Panchal, M. (2014). Review on methods of selecting number of hidden nodes in artificial neural networks. *International Journal of Computer Science and Mobile Computing, 3*(11), 455-464.

Pulvermuller, F. (1999) Words in the brain's language. *Behavioral and Brain Sciences 22*(2), 253–79.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2018). Language models are unsupervised multitask learners. Retrieved from: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

Shafi, I., Ahmad, J., Shah, S.I., & Kashif, F.M. (2006, December 23-24). *Impact of Varying Neurons and Hidden Layers in Neural Network Architecture for a Time Frequency Application*. Paper presented at IEEE International Multitopic Conference, Islamabad. doi: 10.1109/INMIC.2006.358160

Tettamanti, M. et al. (2005). Listening to action-related sentences activates fronto–parietal motor circuits. *Journal of Cognitive Neuroscience, 17,* 273–281.

van der Velde, F., van der Voort van der Kleij, G. T., & de Kamps, M. (2004). Lack of combinatorial productivity in language processing with simple recurrent networks. *Connection Science, 16,* 21–46.

van der Velde, F., & de Kamps, M. (2006). Neural blackboard architectures of combinatorial

structures in cognition. *Behavioral and Brain Sciences, 29,* 37–70.

van der Velde, F., & de Kamps, M. (2010). Learning of control in a neural architecture of grounded language processing. *Cognitive Systems Research, 11,* 93–107.

van der Velde, F., & de Kamps, M. (2015). Ambiguity resolution in a Neural Blackboard Architecture for sentence structure. In Tarek R. Besold & Kai-Uwe Kühnberger (eds.), *Proceedings of the KI 2015 Workshop on Neural-Cognitive Integration*, (pp 1-14). Dresden Germany.

Van der Velde, F. (2019). *Neural Blackboard Architecture of Language: In situ representation, grammar and performance.* In preparation.

Vigliocco, G., Warren, J., Siri, S., Arciuli, J., Scott, S., & Wise, R. (2006). The role of semantics and grammatical class in the neural representation of words. *Cerebral Cortex, 16,* 1790–1796

Zuidema, W., & de Boer, B. (2018). The evolution of combinatorial structure in language. *Current Opinion in Behavioral Sciences, 21*, 138-144. Retrieved from: https://www.sciencedirect.com/science/article/pii/S2352154617301298?via%3Dihub .

8. Appendix

Appendix 1: Python code

```
1.  import keras
2.  from keras.models import Sequential
3.  from keras.layers import Dense, Dropout, Activation
4.  from keras.optimizers import SGD
5.  from keras.utils import np_utils
6.  from keras.utils import plot_model
7.  from keras.models import load_model
8.  import csv
9.  import numpy as np
10. import matplotlib.pyplot as plt
11.
12. file = open('cogsys_zinnen.csv')
13. reader = csv.reader(file)
14. next(reader, None) #skipping the header
15.
16. #Reading and preparing data
17. x_raw = []
18. y_raw = []
19. for line in reader:
20.     x_raw.append(line[0:11])
21.     y_raw.append(line[11:25])
22.
23. x_train = np.asarray(x_raw)
24. y_train = np.asarray(y_raw)
25.
26. data_x = x_train[:15] #n-v-n zinnen
27. data_y = y_train[:15]
28.
29. #model with three layers
30. model = Sequential()
31. model.add(Dense(12, activation='relu', input_dim=11))
32. #model.add(Dropout(0.2))
33. model.add(Dense(12, activation='relu'))
34. #model.add(Dropout(0.2))
35. model.add(Dense(14, activation='sigmoid'))
36.
37. sgd = SGD(lr=0.02, decay=1e-6, momentum=0.9, nesterov=True)
38. model.compile(loss='binary_crossentropy',
39.               optimizer='sgd',
40.               metrics=['accuracy'])
41. model.summary()
42.
43. history = model.fit(data_x, data_y, epochs=50000, shuffle=True, verbose=2)
```

```
44.
45.  # summarize history for accuracy
46.  plt.plot(history.history['acc'])
47.  plt.title('model accuracy')
48.  plt.ylabel('accuracy')
49.  plt.xlabel('epoch')
50.  plt.legend(['train', 'test'], loc='upper left')
51.  plt.show()
52.
53.  # summarize history for loss
54.  plt.plot(history.history['loss'])
55.  plt.title('model loss')
56.  plt.ylabel('loss')
57.  plt.xlabel('epoch')
58.  plt.legend(['train', 'test'], loc='upper left')
59.  plt.show()
60.
61.  model.save("phase1.h5")
62.  print('saved model')
63.  del model
64.
65.  # Phase two training
66.  full_model = load_model('phase1.h5')
67.
68.  full_history = full_model.fit(x_train, y_train, epochs=50000, shuffle=True, verbose=2)
69.  full_model.save('full_model.h5')
70.  print('saved model')
71.
72.  # summarize training accuracy for first and second phase training
73.  plt.plot(history.history['acc'])
74.  plt.plot(full_history.history['acc'])
75.  plt.title('model accuracy')
76.  plt.ylabel('accuracy')
77.  plt.xlabel('epoch')
78.  plt.legend(['phase 1', 'phase 2'], loc='lower right')
79.  plt.show()
80.
81.  file = open('test_sentences.csv')
82.  reader = csv.reader(file)
83.
84.  next(reader, None)
85.
86.  #Reading and preparing data
87.  x_raw = []
88.  y_raw = []
```

```
89.  word = []
90.  for line in reader:
91.      x_raw.append(line[0:11])
92.      y_raw.append(line[11:25])
93.      word.append(line[25])
94.
95.  x_test = np.asarray(x_raw)
96.  y_test = np.asarray(y_raw)
97.
98.  score = full_model.evaluate(x_test, y_test)
99.  pred = full_model.predict(x_test)
100. print(score, pred)
```

Appendix two: Encoded training sentences

b, e, Nrc, Ncc, V, C, T, CC, CV,CT,RC, S, -S, NnS, NnC, NtV, VvS, VvC, VtC, CcN,T,CC CV, CT, RC

1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, sees
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, dog
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,0,1,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,1,0,0,0, fact
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, worries
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, cat
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, knows
0,0,0,1,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,1,0,0,0,0, fact
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, that
0,0,0,0,1,0,0,0,1,0,0, 0,0,0,0,0,0,1,0,0,1,0,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, mouse
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, sees
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, dog
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, that
0,0,1,0,0,0,0,0,0,0,1, 0,0,0,1,0,0,0,0,0,0,0,1,1, dog
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,1,1,0,0,0,0,0,0, sees
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, mouse

0,1,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,0,1,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, who
0,0,0,1,0,0,0,0,0,1, 0,0,0,1,0,0,0,0,0,0,1,0,1,0, fact
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,1,1,0,0,0,0,0,0, worries
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,0,1, mouse
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,0,1,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,1,0,0,0, fact
0,0,0,0,0,1,0,1,0,0,0, 0,0,0,0,0,0,0,0,1,0,1,1,0,0, that
0,0,1,0,0,0,0,1,0,0,0, 0,0,0,1,0,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,1,0,0,0,1,0,0, 0,0,0,0,0,0,1,0,0,1,0,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,0,1, mouse
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, worries
0,0,1,0,0,0,1,0,0,0,1, 0,0,0,0,1,0,0,0,0,0,0,0,0,1, dog
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0,0, einde zin

Appendix three: Encoded test sentences

b, e, Nrc, Ncc, V, C, T, CC, CV,CT,RC, S, -S, NnS, NnC, NtV, VvS, VvC, VtC, CcN,T,CC CV, CT, RC

1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, that
0,0,0,0,1,0,0,0,1,0,0, 0,0,0,0,0,0,1,0,0,1,0,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, mouse
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, that
0,0,0,0,1,0,0,0,1,0,0, 0,0,0,0,0,0,1,0,0,1,0,0,0,0, likes
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, boy
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, sees
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, dog
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, sees
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, dog
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, that
0,0,0,0,1,0,0,0,1,0,0, 0,0,0,0,0,0,0,1,0,0,1,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, mouse
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, that
0,0,1,0,0,0,0,0,0,0,1, 0,0,0,1,0,0,0,0,0,0,0,1,1, dog
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, that
0,0,1,0,0,0,0,0,0,0,1, 0,0,0,1,0,0,0,0,0,0,0,1,1, boy
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,1,1,0,0,0,0,0,0, likes
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,1,1,0,0,0,0,0,0, sees
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, mouse
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0, einde zin

1,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,1,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, mouse
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, that
0,0,1,0,0,0,0,0,0,0,1, 0,0,0,1,0,0,0,0,0,0,0,1,1, dog
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, that
0,0,1,0,0,0,0,0,0,0,1, 0,0,0,1,0,0,0,0,0,0,0,1,1, boy
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,1,1,0,0,0,0,0,0, likes
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,1,1,0,0,0,0,0,0, sees
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,0,1,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,1,0,0,0, fact
0,0,0,0,0,1,0,1,0,0,0, 0,0,0,0,0,0,0,0,1,0,1,1,0,0, that
0,0,1,0,0,0,0,1,0,0,0, 0,0,0,1,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, that
0,0,1,0,0,0,0,0,0,0,1, 0,0,0,1,0,0,0,0,0,0,0,1,1, boy
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,1,1,0,0,0,0,0,0, likes
0,0,0,0,1,0,0,0,1,0,0, 0,0,0,0,0,0,1,0,0,1,0,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, mouse
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, worries
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, dog
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,1, dog
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, knows
0,0,0,1,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,1,0,0,0, fact
0,0,0,0,0,1,0,1,0,0,0, 0,0,0,0,0,0,0,0,1,0,1,1,0,0, that
0,0,1,0,0,0,0,1,0,0,0, 0,0,0,1,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, that
0,0,1,0,0,0,0,0,0,0,1, 0,0,0,1,0,0,0,0,0,0,0,1,1, boy
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,1,1,0,0,0,0,0,0, likes
0,0,0,0,1,0,0,0,1,0,0, 0,0,0,0,0,0,1,0,0,1,0,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,1, mouse

0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, who
0,0,0,1,0,0,0,0,0,0,1, 0,0,0,1,0,0,0,0,0,0,1,0,1,0, fact
0,0,0,0,0,1,0,1,0,0,0, 0,0,0,0,0,0,0,0,1,0,1,1,0,0, that
0,0,1,0,0,0,0,1,0,0,1, 0,0,0,1,0,0,0,0,0,0,0,0,1,1, boy
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,1,0,0,1,0,0,0,0, likes
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,0,1, dog
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,1,1,0,0,0,0,0,0, worries
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,0,1, mouse
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0,0, einde zin
1,0,0,0,0,0,0,0,0,0,0, 1,0,0,0,0,0,0,0,0,0,0,0,0,0, start zin
0,0,1,0,0,0,0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,0,1, cat
0,0,0,0,1,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,1,0,0,0,0, chases
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,0,1, mouse
0,0,0,0,0,1,0,0,0,0,1, 0,0,0,0,0,0,0,0,1,0,0,1,0,1, who
0,0,0,1,0,0,0,0,0,0,1, 0,0,0,1,0,0,0,0,0,0,1,0,1,0, fact
0,0,0,0,0,1,0,1,0,0,0, 0,0,0,0,0,0,0,0,1,0,1,1,0,0, that
0,0,1,0,0,0,0,1,0,0,0, 0,0,0,1,0,0,0,0,0,0,0,0,0,1, boy
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,1,0,0,1,0,0,0,0, likes
0,0,1,0,0,0,1,0,0,0,0, 0,0,0,0,1,0,0,0,0,0,0,0,0,1, dog
0,0,0,0,1,0,0,0,1,1,0, 0,0,0,0,0,0,0,1,1,0,0,0,0,0, worries
0,1,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,0,0,0,0, einde zin

Appendix three: Input-output relations in the training and testing data.

Training sentences.

Table 4: Input-output sequence for sentence Fig 3: *cat sees dog.*
(BB= blackboard).

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
cat	Nrc		N-n-S, RC	RC
sees	V		V-v-S, T	T
dog	Nrc	T	N-t-V, RC	RC
	E		-S	

Table 5: Input-output sequence for sentence Fig 4a: *fact worries cat.*
(BB= blackboard).

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
fact	Ncc		N-n-S, CC	CC
worries	V		V-v-S, T	T
cat	Nrc	T	N-t-V, RC	RC
	E		-S	

Table 6: Input-output sequence for sentence Fig 4b: *cat knows fact.*
(BB= blackboard).

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
cat	Nrc		N-n-S, RC	RC
knows	V		V-v-S, T	T
fact	Ncc	T	N-t-V, CC	CC
	E		-S	

Table 7: Input-output sequence for sentence Fig 5: *cat that chases mouse sees dog.* (BB= blackboard).

Word	Input word node	Input conditional node (BB)	Output node	Active conditional note (BB)
	B		S	
cat	Nrc		N-n-S, RC	RC
that	C	RC	C-c-N, CV, RC	CV, RC
chases	V	CV	V-v-C, T	T
mouse	Nrc	T	N-t-V, RC	RC
sees	V		V-v-S, T	T
dog	Nrc	T	N-t-V, RC	RC
	E		-S	

Table 8: Input-output sequence for sentence Fig 6a: *cat that dog sees chases mouse.* (BB= blackboard).

Word	Input word node	Input conditional node (BB)	Output node	Active conditional note (BB)
	B		S	
cat	Nrc		N-n-S, RC	RC
that	C	RC	C-c-N, CV RC	RC, CV
dog	Nrc	RC	N-n-C, RC, CT	RC, CV, CT
sees	V	CV, CT	V-v-C, V-t-C	
chases	V		V-v-S, T	T
mouse	Nrc	T	N-t-V, RC	RC
	E		-S	

Table 9: Input-output sequence for sentence Fig 6b: *cat who fact worries chases mouse.* (BB= blackboard).

Word	Input word node	Input conditional node (BB)	Output node	Active conditional note (BB)
	B		S	
cat	Nrc		N-n-S, RC	RC
who	C	RC	C-c-N, CV RC	CV, RC

fact	Ncc	RC	N-n-C, CC CT	CV, CC, CT
worries	V	CV, CT	V-v-C, V-t-C	
chases	V		V-v-S, T	T
mouse	Nrc	T	N-t-V, RC	RC
	E		-S	

Table 10: Input-output sequence for sentence Fig 7: *fact that cat chases mouse worries dog.* (BB= blackboard).

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
fact	Ncc		N-n-S, CC	CC
that	C	CC	C-c-N, CV CC	CV, CC
cat	Nrc	CC	N-n-C, RC	CV, RC
chases	V	CV	V-v-C, T	T
mouse	Nrc	T	N-t-V, RC	RC
worries	V		V-v-S, T	T
dog	Nrc	T	N-t-V, RC	RC
	E		-S	

Test sentences

Table 1: Input-output relations for test sentence Fig 8a: Cat that chases mouse that likes boy sees dog.

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
Cat	Nrc		N-n-S, RC	RC
that	C	RC	C-c-N, CV, RC	CV, RC
Chases	V	CV	V-v-C, T	T
Mouse	Nrc	T	N-t-V, RC	RC
That	C	RC	C-c-N, CV, RC	CV, RC
Likes	V	CV	V-v-C, T	T
boy	Nrc	T	N-t-V, RC	RC
Sees	V		V-v-S, T	RC, T
Dog	Nrc	T	N-t-V, RC	RC
	E		-S	

Table 2: Input-output relations for test sentence Fig8.b: Cat sees dog that chases mouse.

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
Cat	Nrc		N-n-S, RC	RC
Sees	V		V-v-S, T	RC, T
Dog	Nrc	T	N-t-V, RC	RC
That	C	RC	C-c-N, CV, RC	CV, RC
Chases	V	CV	V-v-C, T	CV, T
Mouse	Nrc	T	N-t-V, RC	RC
	E		-S	

Table 3: Input-output relations for test sentence Fig 9a: Cat that dog that boy likes sees chases mouse.

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
Cat	Nrc		N-n-S, RC	RC
that	C	RC	C-c-N, CV RC	CV, RC
Dog	Nrc	RC	N-n-C, RC, CT	RC, CV, CT

that	C	RC	C-c-N, CV, RC	RC, CV, CT
boy	Nrc	RC	N-n-C, RC, CT	RC, CV, CT
Likes	V	CV, CT	V-v-C, V-t-C	CV, CT
sees	V	CV, CT	V-v-C, V-t-C	
chases	V		V-v-S, T	T
mouse	Nrc	T	N-t-V, RC	RC
	E		-S	

Table 4: Input-output relations for test sentence Fig 9b: cat chases mouse that dog that boy likes sees.

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
Cat	Nrc		N-n-S, RC	RC
chases	V		V-v-S, T	RC, T
mouse	Nrc	T	N-t-V, RC	RC
that	C	RC	C-c-N, CV, RC	CV, RC
dog	Nrc	RC	N-n-C, CT, RC	CT, RC, CV
that	C	RC	C-c-N, CV, RC	CT, RC, CV
boy	Nrc	RC	N-c-N, RC, CT	RC, CT, CV
likes	V	CV, CT	V-v-C, V-t-C	CV, CT
sees	V	CV, CT	V-v-C V-t-C	
	E		-S	

Table 5: Input-output relations for test sentence Fig10a: fact that cat that boy likes chases mouse worries dog.

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
Fact	Ncc		N-n-S, CC	CC
That	C	CC	C-c-N, CV CC	CV, CC
Cat	Nrc	CC	N-n-C, RC	RC, CV
that	C	RC	C-c-N, CV, RC	RC, CV

Boy	Nrc	RC	N-n-C, RC, CT	CV, RC, CT
Likes	V	CV, CT	V-v-C, V-t-C	CV
Chases	V	CV	V-v-C, T	CV, CT
Mouse	Nrc	T	N-t-V, RC	RC
Worries	V		V-v-S, T	T
dog	Nrc	T	N-t-V RC	RC
	E		-S	

Table 6: Input-output relations for test sentence Fig 10b: dog knows fact that cat that boy likes chases mouse.

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
Dog	Nrc		N-n-S, RC	RC
Knows	V		V-v-S, T	T
Fact	Ncc	T	N-t-V, CC	CC
that	C	CC	C-c-N, CV, CC	CV, CC
Cat	Nrc	CC	N-n-C, RC	CV, RC
that	C	RC	C-c-N, CV, RC	CV, RC
boy	Nrc	RC	N-n-C, RC, CT	RC, CV, CT
likes	V	CV, CT	V-v-C, V-t-C	CV
Chases	V	CV	V-v-C, T	T
Mouse	Nrc	T	N-n-C, RC	RC
	E		-S	

Table 7: Input-output relations test sentence Fig 11a: Cat who fact that boy likes dog worries chases mouse.

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
Cat	Nrc		N-n-S, RC	RC
Who	C	RC	C-c-N, CV RC	CV, RC

Fact	Ncc	RC	N-n-C, CC, CT	CC, CV, CT
That	C	CC	C-c-N, CV, CC	CC, CV, RC, CT
Boy	Nrc	CC, RC	N-n-C, RC CT	CV, RC, CT
likes	V	CV, CT	V-v-C, T	CV, CT, T
Dog	Nrc	T	N-t-V, RC	CV, CT, RC
worries	V	CV, CT	V-v-C, V-t-C	
Chases	V		V-v-S, T	T
Mouse	Nrc	T	N-v-T, RC	RC
	E		-S	

Table 8: Input-output relations for test sentence Fig11b: Cat chases mouse who fact that boy likes dog worries.

Word	**Input word node**	**Input conditional node (BB)**	**Output node**	**Active conditional note (BB)**
	B		S	
Cat	Nrc		N-n-S, RC	RC
chases	V		V-v-S, T	T
Mouse	Nrc	T	N-t-V, RC	RC
Who	C	RC	C-c-N, CV, RC	RC
Fact	Ncc	RC	N-n-C, CC, CT	CV, CC, CT
that	C	CC	C-c-N, CV, CC	CC, CV, CT
Boy	Nrc	CC	N-n-C, RC	CV, RC
Likes	V	CV, CT	V-v-C, T	T
dog	Nrc	T	N-t-V, RC	RC
worries	V	CV, CT	V-v-C, V-t-C	
	E		-S	

Table 9: Example of array representation of binding symbols of sentence structure Fig8a: Cat that chases mouse that likes boy sees dog.

Input													**Output**													
Word	**B**	**E**	**Nrc**	**Ncc**	**V**	**C**	**T**	**CC**	**CV**	**CT**	**RC**		**S**	**-s**	**Nns**	**Nnc**	**Ntv**	**Vvs**	**Vvc**	**Vtc**	**Ccn**	**T**	**Cc**	**Cv**	**Ct**	**RC**
B	1	0	0	0	0	0	0	0	0	0	0		1	0	0	0	0	0	0	0	0	0	0	0	0	0
Cat	0	0	1	0	0	0	0	0	0	0	0		0	0	1	0	0	0	0	0	0	0	0	0	0	1
That	0	0	0	0	0	1	0	0	0	0	1		0	0	0	0	0	0	0	0	1	0	0	1	0	1
chases	0	0	0	0	1	0	1	0	1	0	0		0	0	0	0	0	0	1	0	0	1	0	0	0	0
mouse	0	0	1	0	0	0	1	0	0	0	1		0	0	0	0	1	0	0	0	0	0	0	0	0	1
that	0	0	0	0	0	1	0	0	0	0	1		0	0	0	0	0	0	0	0	1	0	0	1	0	1
likes	0	0	0	0	1	0	0	0	1	0	0		0	0	0	0	0	0	1	0	0	1	0	0	0	0
boy	0	0	1	0	0	0	1	0	0	0	0		0	0	0	0	1	0	0	0	0	0	0	0	0	1
sees	0	0	0	0	1	0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	1	0	0	0	0
dog	0	0	1	0	0	0	1	0	0	0	0		0	0	0	0	1	0	0	0	0	0	0	0	0	1
E	0	1	0	0	0	0	0	0	0	0	0		0	1	0	0	0	0	0	0	0	0	0	0	0	0