Department of Computer Science,
University of Otago

Technical Report OUCS-2006-02

A model of the relationship between language and sensorimotor cognition

Author:

Joost van Oijen

(visiting student from the University of Twente)

Department of Computer Science,
University of Otago, PO Box 56, Dunedin, Otago, New Zealand

http://www.cs.otago.ac.nz/research/techreports.html

A model of the relationship between language and sensorimotor cognition

Joost van Oijen

December 12, 2005

Abstract

The goal of this report is to simulate how a child learns a language. First children learn single words, later they begin to associate events they see with sentences describing the events. They accomplish this with the help of a tutor, usually the parent. Here the relationship between the perception of an event and a sentence describing the event is investigated. Knott's hypothesis about event perception is used resulting in a sensorimotor model where event perception is described as a sequence of sensorimotor items. The theory of Minimalism is used to map sentences onto an underlying syntactic structure. Knott's proposal is that there is a direct mapping of the sensorimotor items to the items in the syntactic structure. This mapping is explored using neural networks.

Contents

Chapter 1

Introduction

The aim of this paper is to suggest a solution for the relationship between natural language and sensorimotor cognition with the goal of simulating how a child learns a language. According to the theory of cognitive development first developed by Jean Piaget, the sensorimotor stage (0-2 years) is the first stage of cognitive development, which marks the development of essential spatial abilities and understanding of the world. It is at the end of this stage that children learn to associate single words they hear frequently with the meaning of the words. In the next stage children begin to associate events they see with sentences describing the events. They achieve this with the help of a tutor. For example, when a child perceives a physical event, eg. 'The dog follows the man', which is simultaneously described by an adult, the child receives two representations. The perception of the event generates sensorimotor representations while the spoken sentence describing the event consists of a sequence of words. There must be a relationship between these two sequences, after all, children in a later stage are able to describe perceived events by generating sentences they have never heard before. The main goal here is to investigate the relationship between these two representations using the hypothesis of Knott.

The hypothesis of Knott is that the syntactic structure of a sentence describing a physical event can be understood as an encoding of the sensorimotor processes which occur in an agent witnessing the event. In Knott's model of the perception and execution of actions, these sensorimotor processes occur in a characteristic sequence in different areas of the brain. Thus a cognitive representation of an event can be seen as a sequence of distinct sensorimotor states. Such sequences are stored in episodic memory

representations, which preserve their sequential structure. The hypothesis is that these memory representations map onto the underlying 'deep' syntactic structure of sentences.

However, sentences describing a particular sensorimotor event can have a different grammar and word order in different languages. The mapping from the 'deep' syntactic structure (which is the same for all languages) to the 'surface' structure (which differs from language to language) must be learned by a child. The goal of this project is to find a way to learn this mapping through the use of a neural network.

The neural network will be used to learn single words and some notion of syntax. The trained network will then be tested on events represented by sensorimotor sequences that were not in the training data to see if it is able to produce a correct sentence describing the event. Different networks can be trained to learn sentences in different word orders, simulating different languages. For these networks the same sensorimotor inputs can be used because the order of the sensorimotor sequences is thought to be universal to every agent.

The outline of the report is as follows. In chapter 2 a literature review is given of topics that reflect different aspects of the problem. A detailed description of the theory behind Knott's hypothesis is given including the representation of semantics as a sensorimotor sequence and the syntactic structure of sentences using Chomsky's theory of Minimalism. Then the 'division of labor' principle will be outlined which is about the contribution of semantics and syntax to the generation of sentences. Finally a review of the Elman neural network is given along with a justification why this network architecture is a good choice for the problem at hand. In chapter 3 several implemented neural networks will be described and evaluated that simulate some language processing aspects that are useful for the problem. These include a network design which gives a promising solution to the problem of mapping sensorimotor sequences to their corresponding sentences. Chapter 4 consists of some extensions or modifications to the basic network design to handle different some aspects of natural language which go beyond basic word ordering issues. These aspects include the use of idioms and inflections.

Chapter 2

Literature review

2.1 Relationship between language and sensorimotor cognition

In this section Knott's hypothesis of the relationship between natural language and sensorimotor cognition will be presented. In section 2.1.1 Knott's model of representing semantics as a sequence of sensorimotor processes will be described. Section 2.1.2 will outline the model of syntactic structure of sentences using Chomsky's theory of Minimalism. A relationship between the two models will be given in section 2.1.3.

2.1.1 Model of human sensorimotor cognition

Knott's sensorimotor model is about the cognitive processes involved in perceiving a simple transitive event. Knott's proposel is that a transitive action is cognitively encoded as a sequence of attentional operations in which the agent, patient and action occur in a characteristic position.

Decomposition of event perception into sub-processes

In the sensorimotor model, visual perception of an event invokes several relatively separate sub-processes. This is supported by the well accepted hypothesis that the brain's visual processing is organized in separate pathways (e.g. Milner & Goodale, 1995). Each pathway extracts different information from the visual input. For example, one pathway categorises objects according

to their form, another delivers representations about the motor affordances of object, another delivers objects and actions derived from motor information and another delivers information about the most salient object in the environment.

Event perception

Humans perceive the world via a sequence of separate snapshots, each resulting from an eye fixation. Each snapshot delivers a **deictic representation**, a transitory representation of what the eye is currently looking at. Each visual pathway can compute its own deictic representation to encode. Knott's suggestion is that event perception can be decomposed into a strict sequence of deictic representations, which constitute a **deictic routing** (Knott, 2005).

To give an example of such a deictic routine, consider the perception of the transitive event: 'The sealion chases the surfer'.

At **stage 1**, the observer is in an attentional state where objects in the world compete for the observer's attention.

At **stage 2**, the observer selects an object to attend to. This initial attented to object will be the agent of the action, which is the sealion in this example.

At **stage 3**, the observer creates a new attentional environment, centred on the attended-to-object (the sealion). This biases attention to objects which are close to the agent. In this new environment, these objects compete for the observer's attention.

At **stage 4**, the observer selects one of these objects to attend to. This object will be the patient of the action, which is the surfer in this example.

At **stage 5**, the observer is in an attentional state where several possible actions compete for selection. These actions are represented as motor goals.

At **stage 6**, one of these motor actions is selected (chase). As a side effect, the observer once more attends to the agent (the sealion).

At **stage 7**, When the goal motor state is reached, the observer once more attends to the patient of the action (the surfer).

Episodic memory

The goal of the project is to link the 'deep' underlying syntactic structure of a sentence to the sensorimotor system of humans. But sentence generation and event perception are not parallel processes. A sentence is not necessarily

pronounced while perceiving the event. A sentence can be generated long after the event was perceived. The hypothesis is that the syntactic structure of a sentence can be seen as a trace of an episodic memory operation which rehearses the original sensorimotor experience.

In Knott's model, when an event is perceived, the deictic representations are stored in a specific sequence in episodic memory. When this event is recalled, the sequence in which the objects and actions were originally perceived is reactivated. It is this reactivation of the sequence which is linked to sentence creation.

2.1.2 Model of human language

The syntactic model Knott uses for his hypothesis is the theory of Minimalism (Chomsky, 1995), which is the successor of Government-and-Binding (GB) theory (Chomsky, 1981). This theory was chosen because it supports the assumption of universal grammar, which means that at some level of abstraction, the linguistic structure of transitive sentenses must be the same in any language. This is a good model to use for the claim Knott wants to make that the structure of a transitive sentence is an encoding of sensorimotor processes -which are assumed to be universal- involved in an event corresponding to the sentence.

Minimalism

A syntactic theory should provide two things. It should provide a method for identifying the entire set of syntactically well-formed sentences in a language and a method that identifies the meaning of each of those well-formed sentences. Because such a theory cannot enumerate every possible sentence, it needs some general principles. According to these principles a sentence can be broken up into phrases. The well-formedness of each phrase can be considered and an infinite set of sentences can be formed by combining phrases through the use of recursion.

The Minimalism theory comprises two components: a representation of phrases and a generative mechanism used for forming and altering phrases. The generative mechanism is used to explain how the full range of well-formed sentences is produced by combining simple phrases. The phrases are, like most syntactic theories, simple trees so that sentences have a tree-shaped syntactic structure.

In Minimalism, two levels of sentence structure are invoked: phonetic form (PF) and logical form (LF). LF is the syntactic representation of the meaning of a sentence, while PF is the representation of its surface form. LF and PF are both tree-based structures produced by the generative mechanism. The generative mechanism starts with a series of phrase-formation operations to create a basic tree-structure corresponding to a sentence. After that a series of movement operations are performed on the resulting tree-structure. Two sequences of movement operations are performed. First a sequence of overt movement operations is executed, which results in a structure representing the phonetic form of the sentence. The point after the overt movement operations is called the 'spell-out'. This is the point at which the phonetic form is 'read off'. A sequence of covert movement operations is then performed, resulting in the final logical form of the sentence. The full sequence of operations performed on the sentence is called a derivation. Figure 2.1 shows how the generative mechanism of Minimalism works.

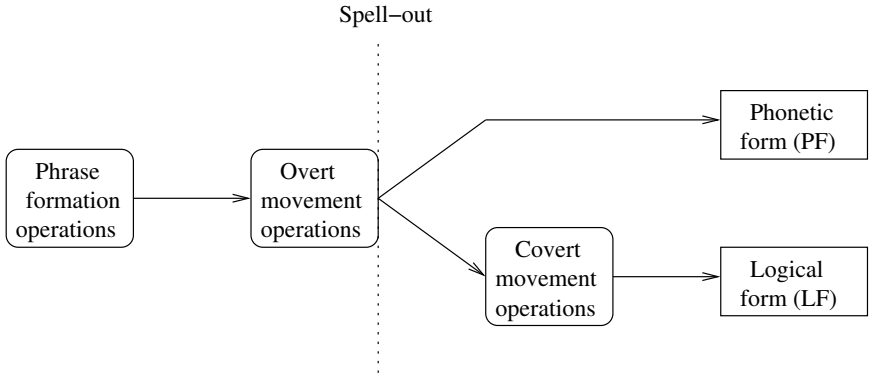

Figure 2.1: Derivations in Minimalist syntax

A theory of phrase structure

The X-bar theory, introduced by Jackendoff (1977), is one of the most used components of theories of grammar, representing syntactic structure of sentences in a tree-based fasion. The key idea is that at some level of abstraction, all phrases have the same structure. The basic element in a phrase is the head, consisting of a lexical item which can be a noun, verb or an adjective for example. The head creates a grammatical constituent, represented by an

X-bar schema, which has certain slots that can be filled by other elements, namely a complement (YP) and a specifier (Spec). The basic X-bar schema is shown in figure 2.2 together with an example of a verb phrase.

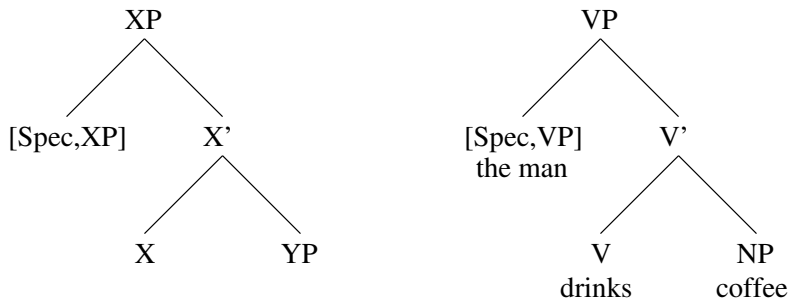

Figure 2.2: The X-bar schema and an schema instance of a verb phrase

X is the head of the phrase, which can be a lexical head (noun, verb or adjective) or a functional head (for instance a verb inflection). YP is the complement of this head and stands for 'any maximal projection'. This means that the X-bar schema is recursive and that a maximal projection has slots which can be filled by other maximal projections. How two X-bar schemas can be bound together is shown in figure 2.3.

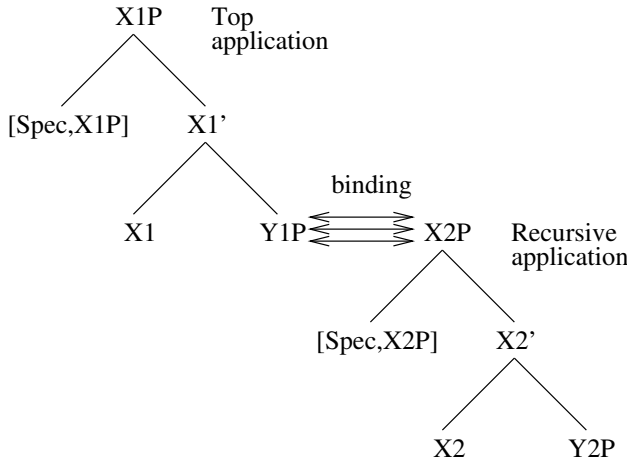

Figure 2.3: A recursive application of the X-bar schema

Clause structure at LF

As an example of how the generative mechanism of Minimalism works, a derivation of the simple transitive sentence 'the man drinks coffee' will be outlined. The derivation begins with a series of phrase formation operations. The X-bar clause structure resulting from these operations is shown in at the top of figure 2.4.

As can be seen, the figure consists of three separate X-bar schemas that are bound together. The bottom projection consists of the verb phrase (VP) which now contains the subject, verb and object of the sentence. The dashed arrows in the figure represent the movement operations that can be performed by the generative mechanism to form the LF of the sentence. The bottom of figure 2.4 shows the structure after all the movement operations, which results in the LF of the sentence.

The tree starts with an inflection phrase (IP). Inflection phrase is the general term used for 'sentence'. The left-hand daughter of IP ([Spec,IP]) is associated with the subject of the sentence. The head of the VP projection moves in two stages to join the I node, the head of the IP. Noun phrases have been relabelled as determiner phrases (DP). Determiners are the heads of the phrases representing the subject and object positions in the sentence structure. An extra XP can be seen between the IP and VP projections. This XP is called the agreement phrase (AgrP). Chomsky proposed that the object DP should raise from its original position as the complement of V to [Spec,AgrP], as can be seen from the dotted arrow. Also, when V moves to its inflection in the SVO-language, it passes through the head of AgrP.

Minimalism says that the different word orders of different languages are caused by the movement operations, which can occur before or after spell-out. Remember from the previous section that the PF of the sentence is 'readoff' after the overt movements and that further covert movements result in the LF. We will now consider two derivations, one for the given sentence in a SVO-language (English, French, Dutch) and one in a SOV-language (Japanese, Korean). Figure 2.5 shows the possible movements caused by the generative mechanism. The difference between the word order of languages has to do with after which movement the PF is 'readoff'. So in an SVO-language, like English, the PF is 'read-off' after movement 1. But in an SOV-language, movements 1 and 2 are overt and so the PF is 'readoff' after these movements, which results in a subject-object-verb word order.

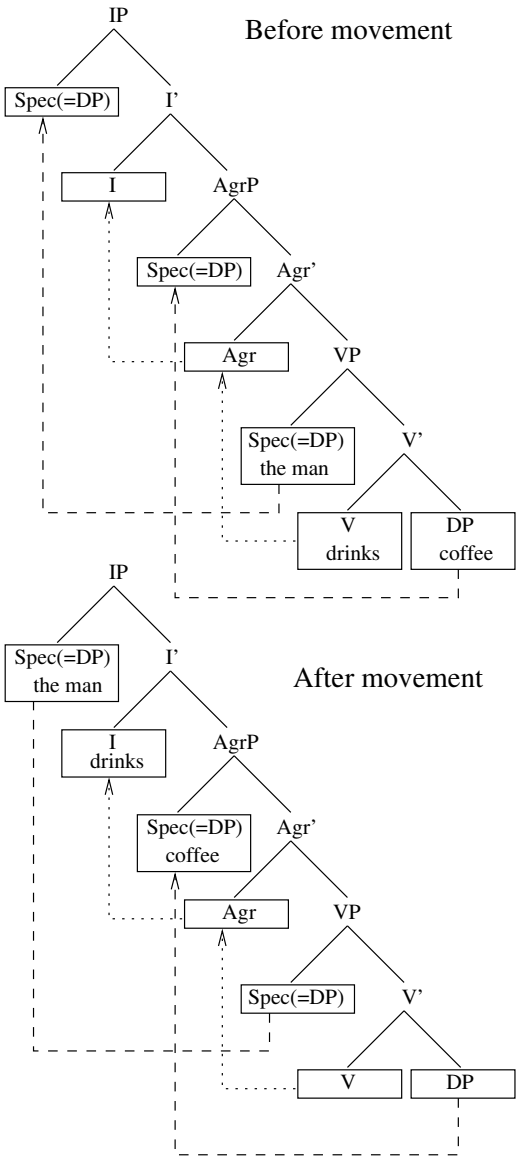

Figure 2.4: X-bar clause structure of a transitive sentence before and after movement

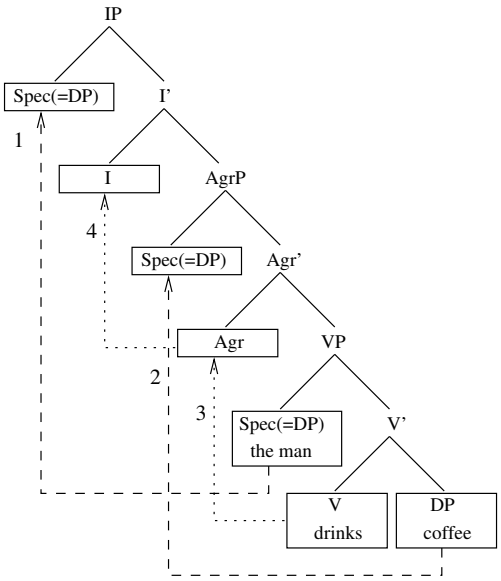

Figure 2.5: The possible movements after phrase-formation operations

2.1.3 Relationship between the models

In this section Knott's hypothesis about the relationship between sensorimotor and syntactic representations will be outlined. The sensorimotor model of Knott was described in section 2.1.1 and the syntactic model was described in section 2.1.2. Knott's proposal is as follows: "The LF of a sentence describing an event encodes the structure of the episodic memory representation of the sensorimotor process via which this event was witnessed". In other words, Knott believes that the LF of a sentence is an encoding of the sensorimotor sequence.

Recall from section 2.1.1 the seven stages of event perception. In figure 2.6 these stages are mapped onto the syntactic structure according to the Minimalism theory using the example sentence 'the sealion chases the surfer'. The figure shows that the order of the sensorimotor representations matches the order at the syntactic level.

So the syntactic structure of Minimalism maps nicely onto the idea of iterative sensorimotor operations. The idea that subject and object DPs are each found in two locations at the LF maps onto the idea that the agent and patient are each attended to twice during the sensorimotor sequence.

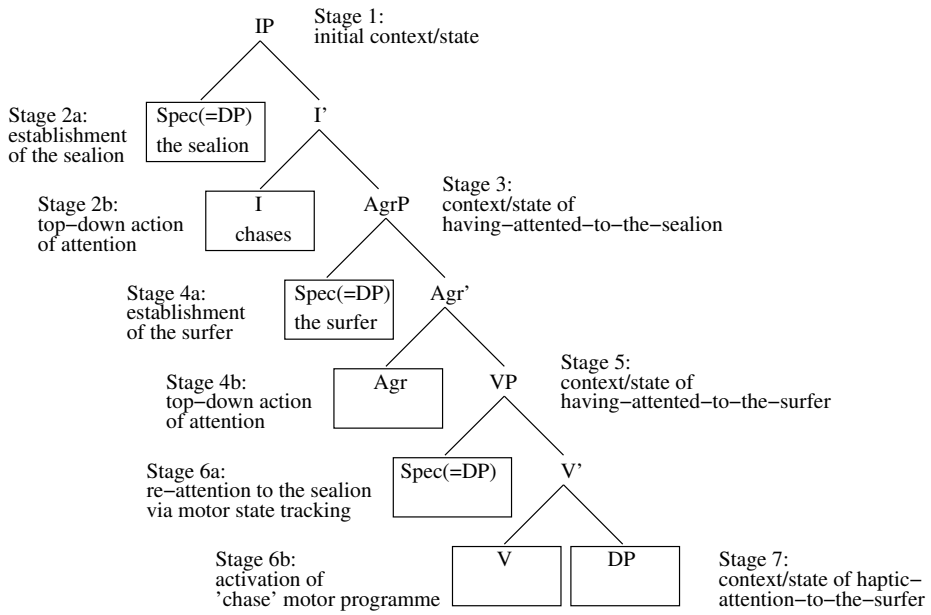

Figure 2.6: Relationship between the sensorimotor model and syntactic structure

2.2 The Elman neural network

Now that the relationship between the syntactic model and the sensorimotor model has been outlined, a neural network model has to be found which can provide a solution for the implementation of this relationship.

A language expresses itself as temporal sequences. This means that a network that handles these temporal sequences needs to have some kind of memory. In his paper 'Structure of Time', Elman suggests the following solution for providing a network with memory. He proposes to augment the network at the input level with additional units, the so called 'Context Units'. These units do not interact with the outside world, but only with other nodes internal to the network. The goal of the context units is to remember the previous state of the network, by copying the weights of the hidden units at each time unit. The hidden units now have the task of mapping both an external input and the previous internal state provided by the context units to some desired output. A network that provides these context units is known as an 'Elman network'.

In his paper Elman applies his architecture to a number of problems that involve processing inputs that have some sequential dependencies which revealed some interesting results. Interesting for the problem at hand are the results of the network that learned structure in sequences, like simple sentences. For example, a network was trained on simple 2, 3-word sentences. After training a hierarchical clustering analysis was performed on the internal representation of the network. It revealed that the network had learned that there are several major categories of words, for example verbs and nouns.

Although the network was not able to predict the precise order of words, it recognizes that there is a class of inputs that typically follow other inputs. The network had learned some kind of syntactic structure. The hidden units of such a network thus not only represent inputs, but also develop representations which will serve as useful encoding of temporal context that can be used when processing subsequent inputs.

2.3 'Division of labor'-principle

One problem with Elman's model is that his neural network model has no semantic inputs. For example, when applying his network to learn small 3-word transitive sentences, it can predict a sentence only by looking at the first

input. When the first input is a noun, the network will most likely predict a verb and after that a noun again because it has learned this syntactic structure. Which verb the network predicts is a matter of chance depending on the frequency of the occurence of a verb after the first noun input. The same holds for the prediction of the last noun.

What is needed is an extra input to the network, a semantic input which can influence the output of the network, so that a sentence can be generated that corresponds to the semantics. For the problem to be solved in this report, the semantics are represented by sensorimotor activations.

How semantics contribute to the generation of sentences is described in the paper 'Learning to divide labor' (Gordon, Dell, 2002). The production of an utterance begins with a message or meaning that one wants to deliver. This message guides the retrieval of lexical items and the construction of a schema that specifies in what position and in what form according to some grammar the lexical items appear. So this lexical retrieval is subject to both semantic and syntactic constraints.

In the paper the hypothesis is explored that the relative strength of the respective contributions of semantics and syntax during lexical access varies among words and this variation can explain certain dissociations on aphasic language production. These variations arise from cue competition, resulting in a 'division of labor'. Cue competition is the tendency for inputs, in this case semantic and syntactic inputs, to compete with each other to control output. Gordon and Dell's idea was not originally expressed within the framework of an Elman network for word-sequencing. Here their idea is translated into this framework.

Assume for example a sentence generation Elman network with an extra semantic input. The syntactic input is represented by a combination of the previous word input and the context units. While testing the network after a succesful training, the semantic input and the syntactic input both contribute to the output of the network. The semantic input to the network presents to the network the meaning of the next word that has to be produced while the syntactic input presents to the network of what class of words the next word that is produced should be, for example a noun or a verb. The goal is to let the network depend enough on both inputs to generate valid sentences with the correct meaning.

Chapter 3

The implementation

In this chapter some implemented neural networks are described and evaluated that contribute to the solution of the problem.

In section 3.1 the first network is presented. This network is a normal Elman network that was implemented to learn sequences. Learning sequences is part of the problem because both semantic and syntax representations consist of sequences of sensorimotor and word items respectively. The Elman network was trained and tested on word sequences of different length. The goal of this phase was to find a good network architecture and the right network parameters for an Elman network to learn sequences of some specified length.

In the second network, outlined in section 3.2, the first network was extended with an extra input, which represented the semantics of the next item to be predicted. The network was trained to predict the next word in a sequence given the previous word of the sequence, the context of the previous word in the sequence and a semantic representation of the word to be predicted. The sequences in the training data were structured sequences with the goal of simulating simple grammar. The expected result is that the network learns a mapping between the semantic input of the word to the actual word to be predicted. To force the network to also learn a notion of grammar and syntax, represented by the previous word and context inputs, some degree of noise was added to the semantic input. The goal was to train the network with different degrees of noise on the semantic input to analyse how much the semantic input and the syntactic inputs contribute to the predicted output of the network in the testing phase.

Finally, a third model was implemented, described in section 3.3, which

goal was to learn the mapping between a sensorimotor sequence and a word sequence. Since the sensorimotor sequence contains two occurences of both agent and patient the task of learning the mapping for a given language is to decide which occurence of the agent and patient must be mapped to a word and which occurences must be mapped to a 'GAP' which represents a silence. This model consists of two separate networks. The first network was used to map sensorimotor representations to their corresponding words. The output of this network was fed into the semantic input of the second network, whose task it was to produce a grammatically correct sentence from a sequence of semantic inputs.

3.1 Initial Elman reimplementation

This section describes an implemented Elman network which learns sequences. At first a problem discription in given in section 3.1.1. In section 3.1.2 a description of the architecture of the network is given. Section 3.1.3 presents the problem to be solved and describes the training process. An evaluation is given in section 3.1.4.

3.1.1 Problem description

The first step in trying to solve the language learning problem is to learn sequences using neural networks. The goal of this phase is to find a neural network architecture that is suitable for this task. The sequences to be learned consist of items represented by letters from the alphabet. Sequences of length five are learned. Variations are made on the number of different sequences and on the number of different items that can occur in a sequence. An item can occur more than once in a sequence.

3.1.2 The Elman network

To learn sequences a common neural network architecture is the Elman recurrent network. An Elman network is a normal multiple feedforward neural network extended with a context layer. This context layer can be seen as just another input to the network. It is added to provide the network with memory. This is accomplished by copying the hidden layer to the context layer at each time step. The network is presented at each time step with an

item in the sequence to be learned and the context in which that item occurs in the sequence. A model of an Elman network is given in figure 3.1.

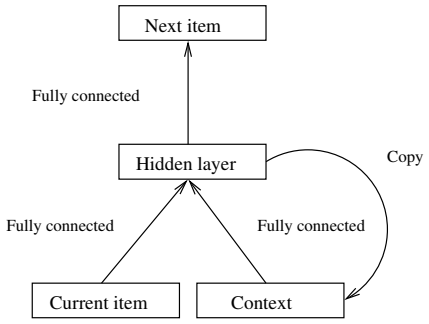

Figure 3.1: An Elman recurrent network

3.1.3 Training the network

The network was trained and tested three times, each time increasing the the amount of sequences and/or the amount of different items that can occur in the sequence. The length of the sequences remained constant, 5. The encoding of a character in the sequence consisted of a bit vector with one bit turned on. A special character was introduced which served as an end-of-sequence character. This character was encoded in the same way as the other characters.

In the training phase subsequent items in a sequence were presented to the network. For each item the prediction error was calculated and the error backpropagated. At the end of a sequence, the context layer was reset to prepare itself for a new sequence. The way the error is backpropagated is shown in figure 3.2.

Training the Elman network consisted of the following steps:

1. Initialize the context layer with random weights
2. Present the first item of the sequence to the current item input layer
3. Calculate the network output at the next item output layer
4. Compare the predicted next item with the actual target item

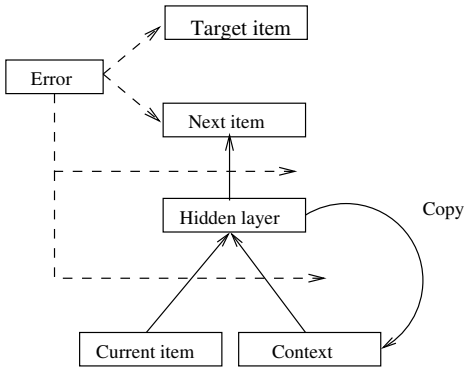

Figure 3.2: Backpropagation in an Elman recurrent network

5. Backpropagate the error by adjusting the weights of the hidden- and output layer

6. Copy the hidden layer to the context layer

7. Repeat steps 2-6, this time by presenting the next item in the sequence until the end of the sequence is reached

8. Repeat steps 1-7 until the training error is sufficiently small

To test the network in the testing phase, one item can be presented to the network. The network predicts the next item that follows the presented item. This predicted item is then fed into the network again as the new current item and the network again predicts the next item. This continues until the network predicts the end-of-sequence character after which the network stops predicting items. If the training data contained sequences that began with the same character, prediction in the testing phase is not always correct because after the first item, it's a matter of chance which next item will be predicted, depending on the number of sequences that began with the same item. To avoid this no sequences with the same first character were trained.

3.1.4 Evaluation

As can be seen from the test results in tables 3.1, 3.2 and 3.3, training a network to learn sequences is a task that can be accomplished fairly easily. The amount of time spent on training though increases rapidly when increasing

parameter	value
sequences:	5
different items:	5
hidden neurons:	15
learning rate:	0.03
momentum:	0.9

training	error	result
1	0.0401	80%
2	0.0401	80%
3	0.0803	60%
4	2.0146 E-4	100%
5	3.3940 E-4	100%

Table 3.1: Test results test 1

parameter	value
sequences:	10
different items:	26
hidden neurons:	20
learning rate:	0.02
momentum:	0.9

training	error	result
1	0.0203	90%
2	0.0403	90%
3	0.0204	90%
4	3.9179 E-4	100%
5	6.4249 E-4	100%

Table 3.2: Test results test 2

the number of sequences to learn and increasing the length of the input bit vector.

3.2 Elman network extended with a semantic input

This section gives an overview of the second network. This network deals with the 'division of labor' principle. A way of representing semantics was needed so the Elman network was extended with an extra input. Section 3.2.1 explains how the 'division of labor' principle will be represented in a neural network. In section 3.2.2 an overview of the network architecture is

parameter	value
sequences:	26
different items:	26
hidden neurons:	30
learning rate:	0.01
momentum:	0.9

training	error	result
1	0.2004	27%
2	0.1005	54%
3	0.0928	62%
4	0.1158	42%
5	0.0928	58%

Table 3.3: Test results test 2

given. Training aspects will be presented in section 3.2.3 and an evaluation is given in section 3.2.4.

3.2.1 Problem description

The 'division of labor' principle is about the sharing of responsibility between syntactic and semantic inputs for lexical activation according to their predictive power. The goal of this phase is to see if this sharing of responsibility can be simulated by a neural network. The syntactic inputs will be represented by the current item input layer and the context input layer. The semantic inputs will be represented by an extra input layer. The items that are fed to this input layer are the semantic representations of the next item that is to be predicted.

As in the previous network, sequences of items are learned. Again the items are represented by letters from the alphabet. The semantic items are represented as upper-case letters. This time, not just random sequences are leared, but sequences of which the order of items are restricted by some underlying structure. This structure consists of rules giving the sequences some grammatical structure. A rule might be for example: "After having seen a 'd' in a sequence, only an 'e' is allowed to follow." The training data for the network then consists of sequences that are created by following the given rules.

The goal is to let the network learn two things. One thing is a mapping of some semantic input representation of the next item in a sequence to the actual next item in the sequence. The other thing is a mapping between the syntactic input representing the current item of the sequence to the next item in the sequence with the use of the learned structure. If the network is trained normally, the network will probably end up having learned sequences only by looking at the semantic information of the next item, because this information provides a direct mapping to the item that is to be predicted. To prevent this situation, some noise is added to the semantic input in the training phase to let the network also learn the structure of the sequences (simulating grammar). With increasing or decreasing the amount of noise in the semantic input, the network can be learned to rely more or syntactic input and context input or to rely more on the semantic input to predict the next item in the sequence.

current item	allowed items as next item
a	c
b	a,c,e
c	a,b,d,e
d	e
e	b,c,d

Table 3.4: Structure of the training data

3.2.2 The network

The network used in this section is based on the Elman network implemented in the previous section. That network was extended with an additional input, representing the semantic input. A model of the resulting network is given in figure 3.3.

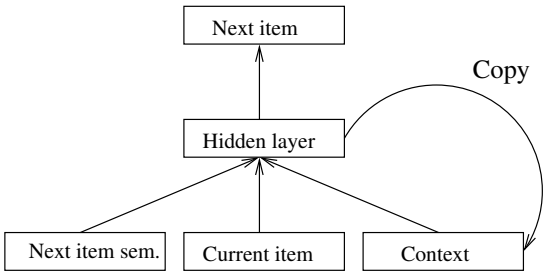

Figure 3.3: An Elman network with an extra semantic input

3.2.3 Training the network

The first task before the network could be trained was to create some structured training data. The data consisted of sequences of length 5, which contained at most 5 different items. The structure that was used for the training data is shown in table 3.4. 500 random sequences were generated as training data that all conformed to the given structure. An example of how one sequence is trained in the network is given in table 3.5.

The error calculation, backpropagation of the error and the resetting of the context after each sequence is the same as in the Elman network of the

step	semantics next item	current item	context	target item
1	E	init	C0	e
2	C	e	C1	c
3	A	c	C2	a
4	C	a	C3	c
5	D	c	C4	d

Table 3.5: Training the network

previous section.

Three tests were performed to analyse the dependence of the network on the semantic and syntactic inputs. In the first test the training data was trained as generated without noise on the semantic inputs. In the second test noise of 1 bit was added to the semantic inputs. This was done by flipping one bit of the input vector on every semantic input of the sequence currently learned. In the third test a 3-bit noise was added to the semantic inputs.

After the training, each resulting network was tested on the same set of inputs. These inputs consisted of sequences of semantic inputs that were randomly generated and thus did not always conform to the structure that was provided in the training data. For testing purposes no noise was added. For every test the output of the network was analysed to see how much the network relied on the semantic inputs and how much it relied on the structure that was learned in the training phase.

3.2.4 Evaluation

The results from the first test were as expected. Without any noise on the semantic output, the network has learned a direct mapping from the semantic input of the next item to the actual next item. So the network primarily relied on the semantic inputs and did not rely on the syntactic structure.

The second test revealed some interesting points. Most of the predicted sequences conformed to the semantic inputs just as in test 1. But in some sequences the network showed that it relied sometimes more on the syntactic and context inputs than on the semantic inputs when predicting the next item. In other words, the network sometimes predicted items that conformed to the learned structure instead of relying only on the semantic next item input. Though the network still relied more on the semantic next item input,

it can be seen that it is possible to learn some of the underlying structure of the sequences when training the network with noise on the semantic inputs.

In the third test the network that was learned with a 3-bit noise on the semantic inputs was tested. The test results showed that the network conformed more to the learned structure than in the previous test. Most of the time the network did not rely on the semantic next item inputs when the structure didn't allow that item to be predicted in the sequence.

The conclusion that can be made from these test results is that one can learn a network to rely more or less syntax (represented by the current item input and the context) by adding noise to the semantics (represented by the next item input) in the training phase. This point is very similar to the 'division-of-labor' point made by Gordon and Dell described in section 2.3.

3.3 Language learning network

In this chapter a network which solves the basic word order learning problem will be discussed. Section 3.3.1 gives a description of the problem to be solved. The network architecture used for this solution is given in section 3.3.2. How this network was trained is presented in section 3.3.3 and a final evaluation of the performance of the network is given in section 3.3.4.

3.3.1 Problem description

The network that we want to achieve is a network that simulates how a child learns a language. Before learning a language, a child first learns single words. So the first task is to simulate this part of the child's development. When a child perceives some object, some sensorimotor activation takes place in his brain. If the child hears a word that corresponds to the perceiving object, the child can learn to create a mapping between this sensorimotor activation and the corresponding word.

The next stage is learning to speak sentences that correspond to some physical event. A cognitive representation of an event can be seen as a sequence of sensorimotor activations. A simplified version of Knott's hypothesis of event perception will be outlined here by giving an example to explain the different sensorimotor activations. Take for example the event of a dog chasing a cat. Knott's suggestion is that a specific sequence of sensorimotor activations takes place in the brain, which is explained in section 2.1.1.

For this model we will use a simplified version of this sequence to see how the network will perform. The sequence of sensorimotor activations assumed here is the following:

1. The observer attends to the dog
2. The observer attends to the cat
3. The observer attends to the dog again
4. The observer attends to the chasing action
5. The observer attends to the cat again

The task to be solved here is to map such sequences of sensorimotor activations to their corresponding sentences, in this case to the sentence: The dog chases the cat. The task will be simplified by learning only the subject, uninflected verb and object of the sentence in question. So the learned sentence of the above example will look like: "dog chase cat". In addition, after having learned alternative sentences, the network should be able to predict new sentences from new sequences of sensorimotor activations that didn't exist in the training data, provided that the network already knows the single sensorimotor activations in the sequence.

The network should also be able to learn to map sequences of sensorimotor activations to sentences of different word orders. The word orders that will be explored here are:

1. Subject-Verb-Object (SVO)
2. Subject-Object-Verb (SOV)
3. Object-Subject-Verb (OSV)

3.3.2 The network architecture

The network solution consists of two independent networks that will be linked together. One network is used for mapping the sensorimotor activations to words corresponding to the activations. This network is a simple feedforward network with one hidden layer. No context layer is needed for this network. The output for this network will be fed as input for the second network.

The second network is used for mapping a sequence of five words to a sequence of three words (the actual sentence). The input of this network is the result of the mapping of the sensorimotor activations to words of the first network. To achieve a mapping of five words to three words a new item is introduced. This item is called a 'GAP'. The purpose of this item is that whenever the network shouldn't produce any output, the GAP item is produced, which is just some representation of a silence. The architecture of the network is the same as the network of the previous section: a normal Elman network with one extra input, the semantic input. The model of the two networks linked together is shown in figure 3.4.

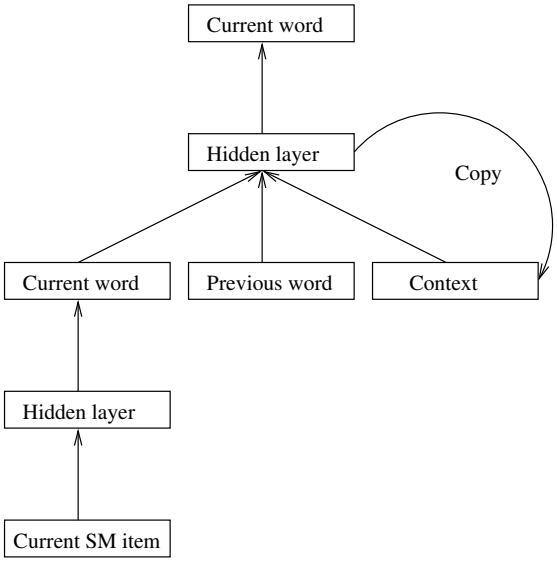

Figure 3.4: Language learning Elman network

3.3.3 Training the network

To train the two networks a lexicon was build that consisted of 20 nouns and 20 transitive verbs. The lexicon is shown in table 3.6.

The first network was trained to map the sensorimotor activation representations of the words in the lexicon to their corresponding words. The training of this network consisted of the usual task of calculating the predic-

nouns				verbs			
man	lion	bird	child	grab	touch	kick	chase
fish	king	weta	dog	see	hear	hit	love
morepork	farmer	penguin	kiwi	defeat	teach	kiss	look_at
teacher	student	surfer	sealion	pick_up	hate	reject	dislike
spider	mother	captain	princess	throw	follow	ignore	greet

Table 3.6: the lexicon

tion error and backpropagating the error. When the overall error was low enough, the network had learned the mapping.

The training of the second network was a bit more complicated. For the training data 500 training pairs of sensorimotor-sequences and word-sequences were generated by combining nouns and verbs from the lexicon. From these 500 items, 20 training items were deleted. These items were reserved for testing the network to see if the it has learned to generalize and can predict sentences from sequences of sensorimotor activations which it hasn't seen in the training phase. The input for the network was sequences which had as format: [subject,object,subject,verb,object]. In the whole network model such sequences are supposed to be the outputs of the first network after mapping the sensorimotor inputs to words sequences. The target items were actual sentences which format depended on the word order used in the training data. In this case we will discuss 'English' word order. The basic idea here is that a word-sequence must be mapped to a sentence in a specific word order. This can be accomplished by inserting GAPs on specific places in the input sequence. For example to create a correct sentence in English from the input sequence, the input sequence has to be mapped to [subject,GAP,GAP,verb,object] because the 'English' word order is SVO. But GAPs cannot be used in the training data, because when a child learns a language, he hears only a sentence and not the GAPs. So the target items used in training the network were sequences like: [subject,verb,object]. The network has to learn where to put the GAPs in the sequence. Training a sequence from the training data consisted of the following steps for each item in the sequence:

1. Present the item to the network

2. Calculate the output of the network

3. Compare the presented item with the target item
4. If the items are equal, go to step 7
5. If the items are unequal, go to step 6
6. Replace the target item with the GAP-item and use the replaced target item as target item for the next item in the sequence
7. Calculate the prediction error
8. Backpropagate the error

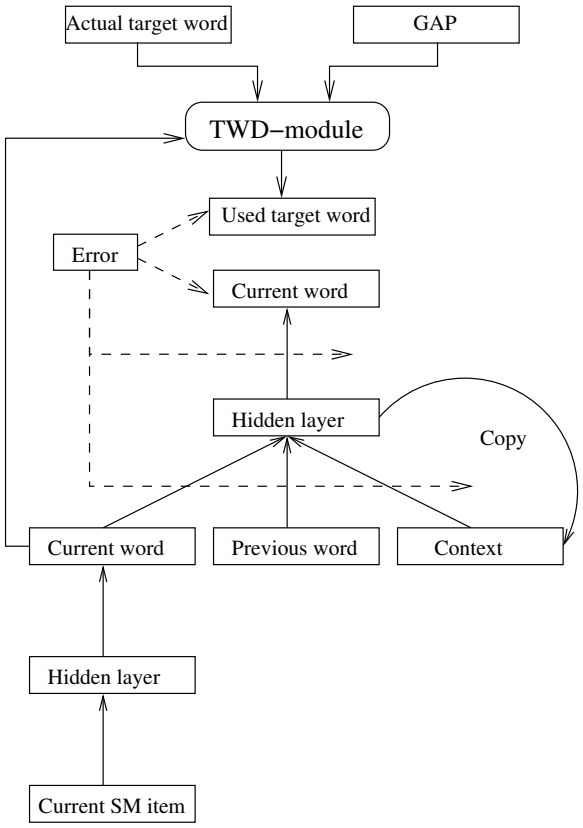

Figure 3.5: Backpropagation in the Language learning network

The network had to learn to produce a GAP whenever the presented item and the target item didn't match. If this was the case, the current target item

had to become the target item for the next presented item. This comparison of items and replacement of the target item is shown in figure 3.5 as the TWD-module (target word decider module). The input to the TWD-module consists of the current item, the target item and the GAP-item. The output is the actual target item wich will be used in the backpropagation algorithm. The output will be the initial target item if that item is equal to the current item, or it will be the GAP-item if the initial target item and the current item are not equal.

After these two networks were trained, the output of the first network (mapping sensorimotor items to words) was connected to the input of the second network (inserting gaps) to form to final network. This network could now be tested by presenting a sequence of sensorimotor items to the 'Current SM item' input shown in figure 3.5. Three different networks were trained like this, one for each word order (SVO, SOV, OSV).

3.3.4 Evaluation

When the network was presented with sequences of sensorimotor items which it had never seen before, the network was still able to predict the corresponding sentences. This was true for all the three networks that learned the different word orders. The test results are shown in table 3.7-3.9

From this can be concluded that it is possible to build a network which simulates the generation of sentences from a sequences of sensorimotor activations. Such a network can be learned to generate sentences with a desired word order, representing different languages, from sensorimotor activations which happen in a specific sequence that is assumed to be universal for every human.

Sensorimotor input	Sentence output
SEALION PRINCESS SEALION HIT SEALION	sealion GAP GAP hit princess
MAN TEACHER MAN LOOK_AT TEACHER	man GAP GAP look_at teacher
STUDENT SPIDER STUDENT DEFEAT SPIDER	student GAP GAP defeat spider
MAN FARMER MAN PICK_UP FARMER	man GAP GAP pick_up farmer
PENGUIN LION PENGUIN TEACH LION	penguin GAP GAP teach lion
SEALION FISH SEALION GRAB FISH	sealion GAP GAP grab fish

Table 3.7: Test results SVO learning network

Sensorimotor input	Sentence output
SEALION PRINCESS SEALION HIT SEALION	sealion princess GAP hit GAP
MAN TEACHER MAN LOOK_AT TEACHER	man teacher GAP look_at GAP
STUDENT SPIDER STUDENT DEFEAT SPIDER	student spider GAP defeat GAP
MAN FARMER MAN PICK_UP FARMER	man farmer GAP pick_up GAP
PENGUIN LION PENGUIN TEACH LION	penguin lion GAP teach GAP
SEALION FISH SEALION GRAB FISH	sealion fish GAP grab GAP

Table 3.8: Test results SOV learning network

Sensorimotor input	Sentence output
SEALION PRINCESS SEALION HIT SEALION	GAP princess sealion hit GAP
MAN TEACHER MAN LOOK_AT TEACHER	GAP teacher man look_at GAP
STUDENT SPIDER STUDENT DEFEAT SPIDER	GAP spider student defeat GAP
MAN FARMER MAN PICK_UP FARMER	GAP farmer man pick_up GAP
PENGUIN LION PENGUIN TEACH LION	GAP lion penguin teach GAP
SEALION FISH SEALION GRAB FISH	GAP fish sealion grab GAP

Table 3.9: Test results OSV learning network

Chapter 4

Extensions

In this chapter a few extensions to the language learning problem are handled. They involve modifications or extensions to the final network solution given in the previous chapter shown in figure 3.4.

First in section 4.1 the 'target word decider' module (TWD-module) will be handled. Recall that the TWD-module was used in the training phase of the language learning network. Its task was to decide what the target word should be for the currently handled word in the sequence. This could be the current target word or a GAP item. The way the TWD-module was implemented cannot be seen as a proper simulation of how the task of the TWD-module would be executed inside the brain. The decision of the actual target word depending on the equality of the current word and the initial target word was done in a programmatic way. To make the simulation of this module more plausible, the TWD-module was implemented in a 'biological' way.

Section 4.2 handles the occurence of idioms in event perceptions and sentences. Take for example the idiom 'keep an eye on'. When perceiving this 'action', a single sensorimotor activation takes place. The current network design can only map this sensorimotor activation to one word. Because the literal item describing the action consists of more that one word, the basic network has to be modified to account for idioms.

Finally, in section 4.3 inflections will be handled. Until now, the word order learning network could only process uninflected verbs. To account for inflected verbs, the network has to be extended with an extra input. This input is a representation of an 'action of attention' to the agent of the event (subject agreement). This input together with the sensorimotor actication

representation input of the action will be mapped to an inflected verb.

4.1 'Biological' implementation of the TWD-module

In this section the 'biological' implementation of the TWD-module will be presented. The goal of the TWD-module is to decide what should be used as target item at each time step of the training phase. This resulting target item can either be the initial target item or a GAP item. The TWD-module has three inputs: the current item (item is that currently presented to the network), the target item for the current item and the GAP item. If the current item and the target item are equal, then the output of the TWD-module should be the target item. If not, the output should be the GAP item.

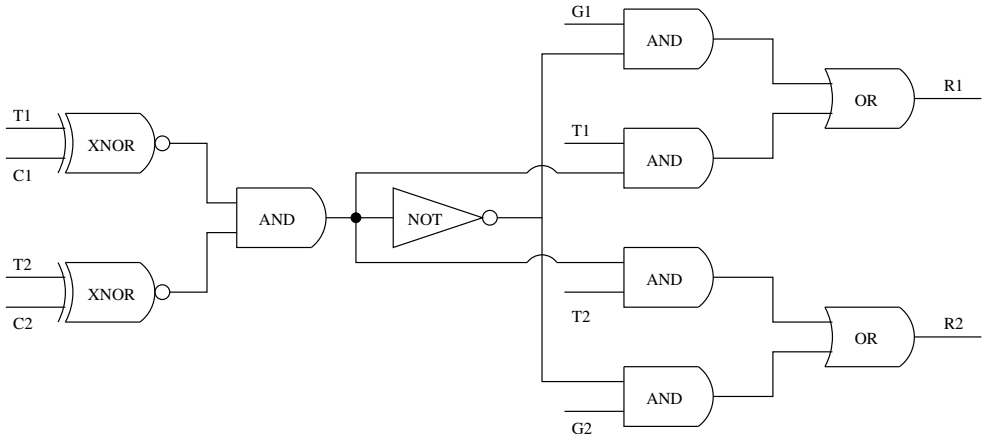

Figure 4.1: Gate circuit of the TWD module

It is assumed that this module in the brain could be hard-wired by evolution. To simulate this, the module was implemented using a circuit of gates. The circuit is shown in figure 4.1. The inputs of the module are bit vectors. To keep it simple bit vectors of only two items are used. So in the figure T1 stands for the first bit of the target item input, C1 for the first bit of the current item input and G1 for the first bit of the GAP item. The outputs of the first XNOR-gates represent the equality of the bit vectors of the target

item and the current item. If the bit vectors are equal, all the outputs of these gates will be 1. The output of the following AND-gate will thus be 1 if the vectors are equals and 0 if they are not. The resulting bit vector (R1..R2) will eventually be the original target item or a GAP item.

4.2 Idioms

In this section the handling of idioms in sentences will be discussed. Section 4.2.1 explains the problem to be solved. The network solution for the problem will be outlined in section 4.2.2. Training the resulting network solution is described in section 4.2.3. The evaluation will be presented in section 4.2.4.

4.2.1 Problem description

An idiom is an expression whose meaning does not follow from the meaning of the individual words of which it is composed. For example the English phrase 'kick the bucket' means 'to die'. Literally, the phrase can in fact mean to kick the bucket, but this is usually not the intended interpretation.

So if our network should learn to handle idioms in sentences, it has to be able to map one sensorimotor activation representation to multiple words. Take for example the following sentence: "You go easy on him". This sentence consists of 5 words, while if one perceives this event, only three different sensorimotor activations take place in the brain. One for 'you', one for 'go easy on' and one for 'him'.

The current final network from section 3.3 cannot handle idioms, because the network can only map one sensorimotor activation representation to one word. To account for idioms the network has to be able to map one sensorimotor activation representation of the idiom to multiple words.

4.2.2 The network architecture

To handle idioms, the part of the final network that maps sensorimotor activation representations to words was modified to handle a mapping of one sensorimotor activation to a sequence of words representing the idiom. In case the input is not an idiom, the output sequence consists just of one word. The part of the final network that learns where to put the GAPs in the sequence did not have to change. The only consequence is that the length of

the sequence of words the network outputs can now be longer than 5.

If the first part of the network now has to learn sequences, an obvious choice is to replace that part of the network with an Elman network. The resulting network architecture is shown in figure 4.2. To connect the two parts of the network, the output of the 'Idiom learning network' is connected to the input of the 'Sentence learning network'.

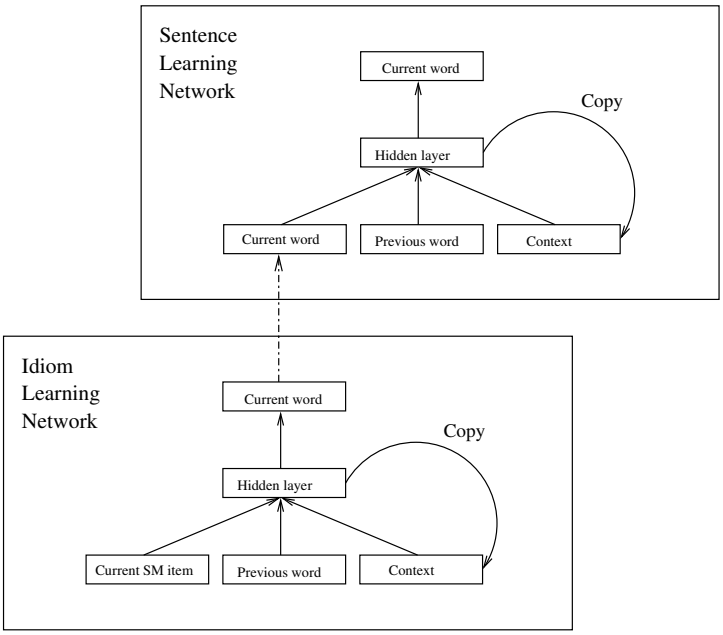

Figure 4.2: Final network extended with idiom extension

4.2.3 Training the network

Just as in the final network, the two parts of the network are trained separately. Part of the training data of the 'idiom learning network' is shown in table 4.1 which shows three idioms and part of the training data of the 'Sentence learning network' is shown in table 4.2. Sensorimotor activation representations of object and actions are represented by the words written in uppercase letters. Remember also that the sensorimotor activations of an event was represented as a sequence of 5: [agent, patient, agent, action, patient].

input	target
MAN	man
SEALION	sealion
BLACK_SHEEP	black sheep
MOTHER	mother
KISS	kiss
HAVE_CRUSH_ON	have crush on
LOSE_TRACK_OF	lose track of

Table 4.1: Training data example of the Idiom Learning Network

input	target
man black sheep man go easy on black sheep	man go easy on black sheep
mother child mother kiss child	mother kiss child
bird fish bird lose track of fish	bird lose track of fish
princess captain princess fall for captain	princess fall for captain

Table 4.2: Training data example of the Sentence Learning Network

When both parts of the network were trained, the final network was build by connecting the output of the first part to the input of the second part of the network.

4.2.4 Evaluation

The trained network solution was able to handle idioms in sentences. The single items in a sensorimotor activation sequence were mapped to a sequence of one or more words, depending on if the item represents an idiom or not. The resulting words of the whole sequence of sensorimotor activations was mapped to a sequence of words including GAP items. Filtering the GAPs from this sequence results in the sentence corresponding to the sensorimotor activations representation of the event that was perceived.

4.3 Inflections

In this section verb inflections will be added to the language learning network of section 3.3. A modification to the sensorimotor model that was used in the language learning network was made. The sensorimotor model that will be used here is described in section 4.3.1. In section 4.3.2 the problem will be outlined. Section 4.3.3 presents the network solution. Training the network is described in section 4.3.4 and an evaluation is given in section 4.3.4.

4.3.1 Modification to the sensorimotor model

In section 3.3.1 a sensorimotor model of event perception was proposed which was represented as a sequence of 5 sensorimotor activation representations in episodic memory. For the solution to the network that handles inflected verbs the model was modified. The model that was used here is shown in figure 4.3 using the example event of a sealion chasing a surfer.

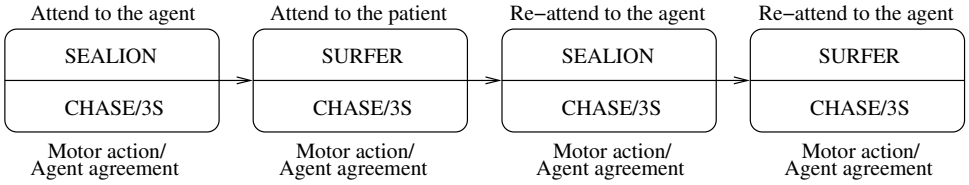

Figure 4.3: Modified sensorimotor model

In this model the length of the sensorimotor sequence is 4 where each item is an attendance to the agent or the patient, combined with the motor action with its agent agreement (causing inflection in sentence generation). Patient agreement will currently be left out as this is not reflected in the verb in most languages.

4.3.2 Problem description

Until now, the network could only learn uninflected verbs.

Knott's proposal is that the inflection of the verb which agrees with the subject is a reflex of the action of attention which results in establishment of the agent. This action representation together with the motor action representation must endure throughout the sequence. The network solution now

has to have three different inputs: A sensorimotor activation representation of an action (verb), the sensorimotor activation representation of the action of attention to the agent (which causes the inflection) and a sensorimotor activation representation of an object (noun). The verb and its inflection endure throughout the sequence. The objects representing the agent and patient of the action change from time step to time step. As an example consider the perception of the event 'the man chases the dog'. The sensorimotor activation sequence of this event perception using the model described above would now look like this: [(MAN,CHASE/3S), (DOG,CHASE/3S), (MAN,CHASE/3S), (DOG,CHASE/3S)]. This sequence has four items and each item consists of a combination of a sensorimotor activation representation of an object and an action with its subject agreement (3S stands for third person singular). A network solution has to be found to map such sequences to their corresponding sentences, in this case the sentence 'man chases dog'.

4.3.3 Network architecture

The network architecture that was used here is shown in figure 4.4. The solution consists of 4 smaller networks. Network 1 maps sensorimotor activation representations of objects to their corresponding nouns. Network 2 maps the combination of sensorimotor activation representations of actions and their subject agreement to the corresponding inflected verbs. Network 3 decides which nouns of the noun input sequence should be a GAP and network 4 decides which verbs of the verb input sequence should be a GAP. At each time step, network 3 and 4 can either produce a noun respectively a verb or a GAP. The inputs of network 3 are the current noun, the previous noun and the context of network 3. The inputs of netword 4 are the current verb, the previous noun and the context of network 3. No context of network 4 is used, but it uses the context of network 3 which copies its hidden layer at each time step to the context inputs.

4.3.4 Training the network

The error calculation and backpropagation in the training phase is shown in figure 4.5. Network 1 and 2 of the figure were trained separately. Error calculation and backpropagation in network 1 and network 2 works normally by calculating the difference between the output and the target item. Network 2

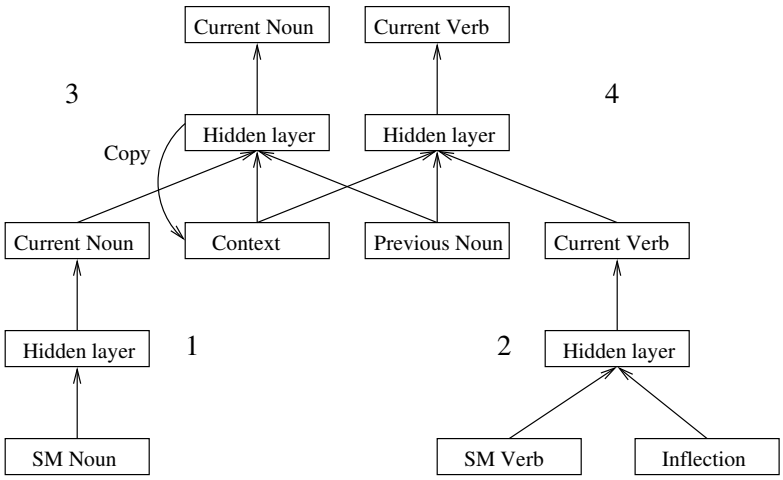

Figure 4.4: Final network extended with inflections

input SM-action	input SM-inflection	output verb
CHASE	1S	chase1S
CHASE	2S	chase2S
CHASE	3S	chase3S
CHASE	1M	chase1M
CHASE	2M	chase2M
CHASE	3M	chase3M

Table 4.3: Training data example of the Idiom Learning Network

handles the mapping of a sensorimotor activation representation of an action and its subject agreement to an inflected verb. The mappings that had to be learned for a single verb are presented in table 4.3. Because inflections for regular verbs in English are not very rich, inflected verbs are represented as the verb followed by the inflection. (It would have been better to use for example Italian as a language in this experiment.)

Network 3 and 4 were trained together because network 4 needs to have the context input of network 3. Error calculation and backpropagating the error in network 3 and 4 work just like in section 3.3.3. Training a sequence in network 3 and 4 is described in the following pseudo-code. The 'nounInputSequence' in the code could be for example [dog, cat, dog, cat] and the

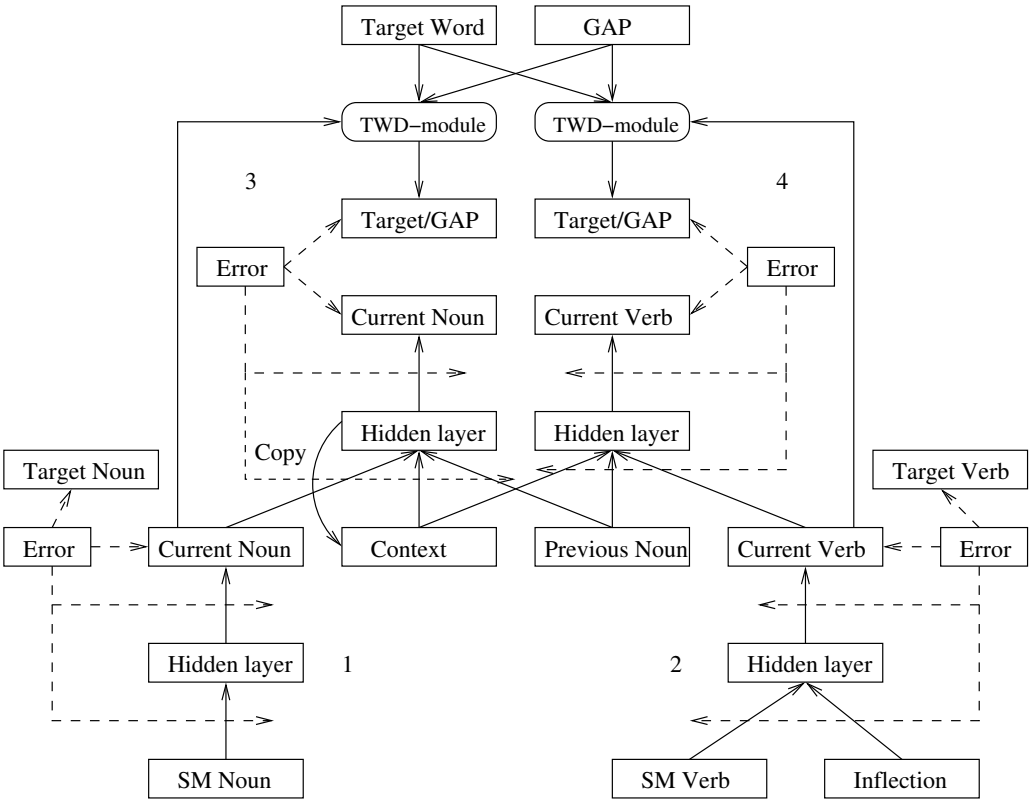

Figure 4.5: Final network extended with inflections

'verbInputSequence' could be [chase3S, chase3S, chase3S, chase3S], then the 'targetWords' sequence would be [dog, chase3S, cat] and the output when testing the networks would be [dog, chase3S, cat, GAP, GAP, GAP, GAP, GAP] (reading the output of network 3 before network 4 at each time unit).

```
train() {
  context3 = initial context
  context4 = context3
  prevWord3 = some initial input representation
  prevWord4 = prevWord3
  targetWordPos = 0

  for i = 0; i < length of the noun input sequence; i++ {
    currentNoun = nounInputSequence[i]
    if targetWords[targetWordPos] = currentNoun {
      targetWord3 = targetWords[targetWordPos]
    } else {
      targetWord3 = GAP
    }
    calculate output network 3 and backpropagate the error
    context3 = weights hidden layer network 3
    if output network 3 is not GAP {
      targetPosition++
    }

    currentVerb = verbInputSequence[i]
    if targetWords[targetWordPos] = currentVerb {
      targetWord4 = targetWords[targetWordPos]
    } else {
      targetWord4 = GAP
    }
    calculate output network 4 and backpropagate the error
    context4 = context3
    if output network 4 is not GAP {
      targetPosition++
    }

    prevWord3 = last output network 3
    prevWord4 = prevWord3
  }
}
```

After training all the parts of the network, the parts were connected. The output of network 1 was connected to the input of network 3 and the output of network 2 was connected to the input of network 4. Four different word orders were trained, namely SVO, SOV, VSO and the less common OSV.

4.3.5 Evaluation

Training the network for the inflected verb sentences was harder than training the network for the uninflected verb sentences. Part of the test results are

shown in tables 4.4-4.6. The predicted GAPs have been left out. Almost all of the tests generated successful sentences. As can be seen from the test results only the second test in the SOV and OSV word orders is wrong. The noun predictions are always correct, but in the second test the verb was predicted wrongly. Although the wrong verb was predicted, the inflection that was predicted was correct, so no grammatical errors were made. So it is possible to train a network to predicted correct sentences with inflected verbs.

input SM-noun	input SM-verb	output
SEALION SURFER SEALION SURFER	(CHASE,3S)	sealion chase3S surfer
I YOU I YOU	(GRAB,1S)	i grab1S you
PENGUIN FISH PENGUIN FISH	(FOLLOW,3S)	penguin follow3S fish
MOTHER TEACHER MOTHER TEACHER	(DEFEAT3S)	mother defeat3S teacher
WE CHILD WE CHILD	(DISLIKE,1M)	we dislike1M chlid

Table 4.4: Test results SVO

input SM-noun	input SM-verb	output
SEALION SURFER SEALION SURFER	(CHASE,3S)	sealion surfer chase3S
I YOU I YOU	(GRAB,1S)	i you follow1S
PENGUIN FISH PENGUIN FISH	(FOLLOW,3S)	penguin fish follow3S
MOTHER TEACHER MOTHER TEACHER	(DEFEAT3S)	mother teacher defeat3S
WE CHILD WE CHILD	(DISLIKE,1M)	we child dislike1M

Table 4.5: Test results SOV

input SM-noun	input SM-verb	output
SEALION SURFER SEALION SURFER	(CHASE,3S)	surfer sealion chase3S
I YOU I YOU	(GRAB,1S)	you i dislike1S
PENGUIN FISH PENGUIN FISH	(FOLLOW,3S)	fish penguin follow3S
MOTHER TEACHER MOTHER TEACHER	(DEFEAT3S)	teacher mother defeat3S
WE CHILD WE CHILD	(DISLIKE,1M)	child we dislike1M

Table 4.6: Test results OSV

input SM-noun	input SM-verb	output
SEALION SURFER SEALION SURFER	(CHASE,3S)	chase3S sealion surfer
I YOU I YOU	(GRAB,1S)	grab1S i you
PENGUIN FISH PENGUIN FISH	(FOLLOW,3S)	follow3S penguin fish
MOTHER TEACHER MOTHER TEACHER	(DEFEAT3S)	defeat3S mother teacher
WE CHILD WE CHILD	(DISLIKE,1M)	dislike1M we child

Table 4.7: Test results VSO

Bibliography

[1] Elman, Jeffrey L., *Finding Structure in Time,* (Cognitive Science 14 pages 179-211).

[2] Gordon, Dell, *Learning to divide the labor: an account of deficits in light and heavy verb production* (Cognitive Science 27 pages 1-40).

[3] Knott, Alistair, *Grounding syntactic representations in an architecture for sensorimotor control* (Technical report OUCS-2003-04).

[4] Knott, Alistair, *Argument linking and spatial cognition*

[5] Goldberg, Adele E., *Constructions: a construction grammar approach to argument structure*

[6] Ullman, Michael T., *Neural correlates of lexicon and grammar: Evidence from the production, reading, and judgement of inflection in aphasia*