Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

Natural Language Processing with Recurrent Neural Networks

2011 RNDr. Ján Švantner

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics
Department of Applied Informatics

Natural Language Processing with Recurrent Neural Networks

Dissertation thesis

RNDr. Ján Švantner

Reference number: a61be154-6582-442f-ac4d-f502124949c9
Programme of study: Informatics
Field of study: 9.2.1 Informatics
Department: Department of applied informatics
Supervisor: doc. Ing. Igor Farkaš, PhD.
Issued: Bratislava, 2011

Univerzita Komenského v Bratislave
Fakulta Matematiky, Fyziky a Informatiky
Katedra Aplikovanej Informatiky

Spracovanie Prirodzeného Jazyka pomocou Rekurentných Neurónových Sietí

Dizertačná práca

RNDr. Ján Švantner

Evidenčné číslo: a61be154-6582-442f-ac4d-f502124949c9
Študijný program: Informatika
Študijný odbor: 9.2.1 Informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: doc. Ing. Igor Farkaš, PhD.
Vydané: Bratislava, 2011

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	RNDr. Ján Švantner
Študijný program:	informatika (Jednoodborové štúdium, doktorandské III. st., denná forma)
Študijný odbor:	9.2.1. informatika
Typ záverečnej práce:	dizertačná
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov : Natural Language Processing with Recurrent Neural Networks

Školiteľ : doc. Ing. Igor Farkaš, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 27.10.2010

Dátum schválenia: 21.04.2011

[illegible]
prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

Švantner
študent

[illegible]
školiteľ práce

Dátum potvrdenia finálnej verzie práce, súhlas s jej odovzdaním (vrátane spôsobu sprístupnenia)

[illegible]
školiteľ práce

Acknowledgements

First of all, I would like to thank my advisor Igor Farkaš for granting me the possibility of doing my Ph.D. study under his supervision. I am very grateful for his guidance and many advises concerning the topic of my thesis.

I would like to thank my family, my girlfriend Zuzana and all friends for supporting me, believing in me and motivating me during my studies.

Abstrakt

Táto dizertačná práca skúma spracovanie prirodzeného jazyka pomocou rekurentných neurónových sietí. Okrem jednoduchej rekurentnej siete (SRN) pre modelovanie používame aj novšie modely, medzi ktoré patria siete s echo stavmi (ESN) a siete s explicitným pozornostným mechanizmom (A-SRN). Dizertačná práca pozostáva z troch hlavných častí. Prvá časť sa venuje modelovaniu nesusedných závislostí medzi slabikami v rámci súvislého reťazca. Overíme hypotézu variability, ktorá tvrdí, že spracovávanie jazyka u ľudí sa opiera o nesusedné závislosti v prípade, že susedné neposkytujú dostatočnú štatistickú informáciu. Ukážeme, že keď je variabilita susedných závislostí príliš vysoká, tak neurónová sieť preferuje použitie nesusedných závislostí. V druhej časti sa zameriame na akvizíciu gramatiky zjednodušených jazykov. Túto úlohu modelujeme pomocou ESN, pričom opíšeme vplyv rôznych vstupných reprezentácií na úspešnosť modelu. Ukážeme, že štatistické vlastnosti vstupných reprezentácií, vytvorených predspracovaním dát zo vstupného jazyka, sú len čiastočne zodpovedné za zvýšenie úspešnosti ESN. Druhou nutnou podmienkou je správna škála vstupných vektorov. V ďalšom texte navrhneme a implementujeme vytvorenie vstupnej reprezentácie, ktorá nesie dostatočné sémantické informácie a je vytvorená pomocou viacerých behov siete s echo stavmi, bez nutnosti použitia iných štatistických metód. Tretia časť dizertačnej práce je venovaná modelovaniu jazykom riadenej pozornosti v rámci vizuálnej scény. Budeme prezentovať viaceré modely s explicitným pozornostným mechanizmom (A-SRN) a porovnáme ich s modelmi SRN a ESN. Preskúmame význam pridanej vizuálnej scény na úspešnosť spracovávania prirodzeného jazyka a ukážeme, že situačný vstup pomáha modelom správne predikovať konkrétne akcie a objekty v opisovanej udalosti. Na druhej strane ukážeme, že jazykový opis upriamuje pozornosť modelu na správne objekty v rámci vizuálnej scény. Výhoda nášho modelu spočíva vo fakte, že umožňuje aj spracovávanie zložitejších scén, čo testujeme použitím dvoch až troch súčasných vstupných udalostí v jednej vizuálnej scéne.

Kľúčové slová: spracovanie prirodzeného jazyka, rekurentné neurónové siete, nesusedné závislosti, akvizícia gramatiky, modelovanie pozornosti

Abstract

This dissertation thesis examines natural language processing using recurrent neural networks. Apart from the standard simple recurrent network (SRN), we also use more recent models like the echo-state network (ESN) and networks with multiple feedback connections which model explicit attentional mechanism (A-SRN-based models). The dissertation thesis deals with three main topics. The first area of interest concerns the non-adjacent dependencies between syllables in continuous syllable sequence. We model the variability hypothesis which claims that human readers can focus on non-adjacent dependencies when the adjacent dependencies do not provide satisfactory statistical information. We show that when the variability of adjacent dependencies is too high, the neural network prefers use of non-adjacent dependencies. In the second part, we concentrate on grammar acquisition of simplified English languages. We model this task using ESN and describe the effect of various input representations on the model performance. We show that statistical properties of input representations, created by preprocessing of the language data, are only partly responsible for the improvement of ESN performance. The other necessary condition is the correct scaling of inputs. Additionally, we propose and implement the generation of reasonable input representation with the multiple executions of ESN, without the need to preprocess the language with statistical methods. In the third part of this thesis, we deal with modelling of the utterance-driven attention within the visual scene. Hence, we focus on language processing not in isolation but rather in the context of the visual information. We present several models with explicit attentional mechanism (A-SRN) and compare them with SRN and ESN. We examine the effect of added visual scene to the performance of language processing and show that the scene input helps the models to correctly anticipate particular actions and objects of the described event. On the other hand, we show that the language drives the model in focusing attention to correct objects from the scene. The benefit of our model resides in the fact, that it can process more complex visual scenes, what was tested using up to three concurrent input events within one visual scene.

Key words: natural language processing, recurrent neural networks, non-adjacent dependencies, grammar acquisition, utterance-mediated attention

Contents

List of Figures

List of Tables

List of Shortcuts

AAN	Auto-associative network
A-SRN	Simple recurrent network with attentional mechanism
BPTT	Back propagation through time
ESN	Echo-state network
RecSOM	Recursive self-organizing network
RTRL	Real-time recurrent learning
SardNet	Sequential activation retention and decay network
SOM	Self-organizing network
SRN	Simple recurrent network

List of Formulas

$\mathbf{v}^\top$	Transpose of a vector $\mathbf{v}$
$.*$	Component-wise multiplication of two vectors
$[\mathbf{u}, \mathbf{v}]$	Concatenation of the vectors u, v.
$d(i, i^*)$	Euclidean distance between units with indices i and i^*
$\|.\|$	Euclidean vector norm
$\lambda(\mathbf{W})$	Spectral radius (the largest absolute eigenvalue) of the matrix $\mathbf{W}$
$\sigma(\mathbf{W})$	The largest singular value of a matrix $\mathbf{W}$
α	Learning rate of the neural network
$P(A\|B)$	Conditional probability
tanh	Hyperbolic tangent function

Dedicated to my family and all my friends.

Chapter 1

Introduction

Language is an important feature of human intelligence. It gives us possibility to communicate, allows us to exchange information and experience, describe surrounding world and make joint decisions (Siskind, 2001). Language markedly separates humans from other animal species.

Over time, two computational approaches of language modelling have emerged. Symbolistic approach uses for language processing set of rules, operating with finite number of symbols. Formal nature of symbolistic approach allows us to elegantly describe various classes of language and prove their properties. It allows also description of complex languages with possibly recurrent structure. Downfall of this approach is, that it is unclear how to map existing real world objects to the symbols and how to choose degree of detail (symbol-grounding problem; Harnad, 1990).

Connectionist (subsymbolic) approach deals with input data using artificial neural networks. Neural network is trained directly on input data using general learning procedures, therefore it does not require task-specific design (i.e. one network can process various data sets). The advantages of subsymbolic language processing approach are:

- **Generalization:** neural networks are able to generalize and adapt to the novel inputs, using their similarities with already learned concepts. Since natural language is very complex and often uses new words and combination of concepts, this ability is crucial for its successful processing.

- **Learning:** connectionist models learn from experience, what enables their improvement also in changing environment. Learning from examples does not require hand-designed architecture and allows reusability of the model for different tasks.

- **Distributed environment:** many neural architectures can be executed within distributed environment what decreases the execution time.

- **Biological plausibility:** in spite of the fact that artificial neural networks are only simplified models of real neural networks, they are more biologically plausible than symbolic systems and therefore allow better understanding of human language processing. Neural networks are often used for modeling human behaviour with its strong and weak properties.
- **Modeling of defects:** people makes lot of mistakes during language processing. In serious cases are the individuals unable to perform certain language-processing tasks. These illnesses are called lesions. Damaging a neural network is often used to model lesions what helps us to understand why they develop and how they can be avoided and cured.

Natural language processing domain consists of various subtasks, which can be split into following categories (Christiansen and Chater, 1999b). On the **lower level**, people need to process single words, detect their boundaries in continuous sequences of characters, learn to comprehend and to pronounce them correctly. Moreover, natural languages have complicated rules and exceptions mechanisms for processing word morphology. Complexity of these tasks is magnified by fact, that children in early stages of life have no direct feedback for development of lexical processing skills. The **higher level** of language processing applies to sentence processing. Most important tasks in this domain are: sentence parsing (syntactic analysis), sentence comprehension and sentence production. Alongside with these main tasks, we observe supportive subtasks like word prediction and grammar acquisition. They are used during sentence production (by iteration of word prediction process) and ease the production and comprehension by accessing correct concepts with a top-down mechanism. During the last decade, research into human language comprehension has begun to examine processing of human language in presence of the visual scene. Visual scene enhances spoken language, providing features of underlying concepts, while on the other side spoken language drives attention to different parts of the scene. Many of the cognitive processes can be assessed using the visual world paradigm, which is an excellent method for studying language, vision, memory and attention and to understand their mutual interplay (Huettig et al., 2011).

In Chapter 2 we describe neural network architectures which are commonly used in natural language processing. In chapter 3 we summarize the current state of the research field for prediction paradigm. In later text, we focus on three of the previously mentioned subtasks. In Chapter 4 we investigate the performance of the echo-state networks (see chapter 2.3) in predicting of non-adjacent dependencies and its ability to detect word boundaries in the continuous sequence of syllables. Chapter 5 uses echo-state networks to process sentences and acquire underlying grammar. Throughout analysis of the effect of input representations is presented in this chapter, taking into consideration localist, random distributed and statistical representations. In the last section of this chapter we present technique to create reasonable input representations with the multiple executions of the echo-state network. In Chapter 6 we model the utterance-driven visual attention with various recurrent neural network architectures, including A-SRN (see Chapter 2.2) which uses an explicit attentional mechanism.

Chapter 2

Neural network architectures

2.1 Simple recurrent network

Simple recurrent neural network (SRN) was developed by Elman (1990) as an extension of the feed-forward neural network. Elman trained SRN on character and word prediction tasks and showed that model is able to learn underlying grammar. Architecture of SRN extends multilayer feed-forward network with additional recurrent layer. State of the hidden layer is copied to the context layer and is presented back in next step via a set of recurrent weights. The activation of simple recurrent networks in time t can be expressed using formulas:

$$\begin{aligned}\mathbf{a}_{\text{hid}}(t) &= G(\mathbf{W}_{\text{in}}\mathbf{a}_{\text{in}}(t) + \mathbf{W}_{\text{hid}}\mathbf{a}_{\text{hid}}(t-1)) \\ \mathbf{a}_{\text{out}}(t) &= F(\mathbf{W}_{\text{out}}\mathbf{a}_{\text{hid}}(t)),\end{aligned}$$

where $G(net)$ is the hidden and $F(net)$ is the output activation function. The most commonly used functions are:

$$f(net) = \begin{cases} net & \text{linear function} \\ \tanh(net) & \text{hyperbolic tangent} \\ \frac{1}{1+e^{-\lambda net}} & \text{unipolar sigmoid} \\ \frac{2}{1+e^{-\lambda net}} + 1 & \text{bipolar sigmoid} \end{cases}$$

SRN can be trained using standard error backpropagation, backpropagation through time (BPTT; Rumelhart et al., 1986) or real-time recurrent learning (RTRL; Williams and Zipser, 1989). The latter two algorithms are more computationally demanding but achieve better results. We have trained SRN with BPTT algorithm which is described in more detail in Chapter 2.2. The latest and most successful training methods use Kalman and the extended Kalman filters. However, these algorithms require increased amount of memory and are computationally very complex, making the training method ineffective for

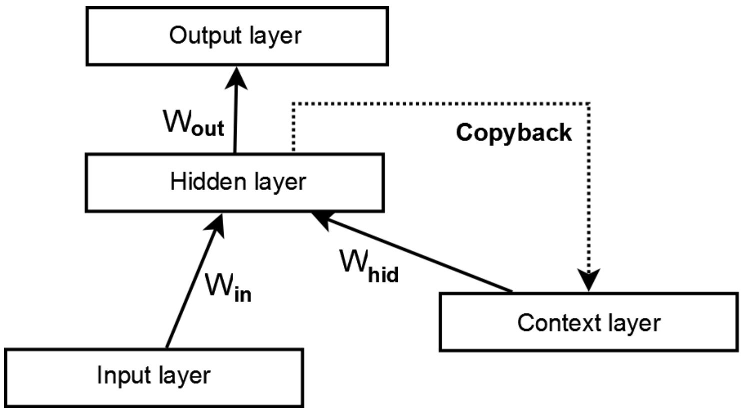


Figure 2.1: Architecture of SRN. Activation of hidden layer is copied to context layer and presented to hidden layer in next step.

tasks which does not require such advanced training procedure or tasks with large data sets.

Recurrent connection allows SRN to successfully process various time series, including signal sequences, melodies, simplified natural language and data generated by context-free grammars. State of recurrent layer represents most current part of input sequence, demonstrating both markovian and non-markovian behaviour. Čerňanský and Makula (2007) have shown that RNN possesses architectural bias and activations of recurrent units exhibit structural differentiation even before training. The amount of structural differentiation of untrained RNN is comparable to variable length Markov models. Čerňanský and Makula (2007) claimed, that fixed point attractors for each input exist even before training but they are randomly placed. During training, attractors change their positions to express characteristics of input data, placing attractors for similar data to the same regions.

2.2 Recurrent network with explicit attentional mechanism

We have introduced simple recurrent network with explicit attentional mechanism (A-SRN) in Švantner et al. (2011b) and further improved it in Švantner et al. (2011a). As the name suggests, the network architecture is extended with extra connection, which is designed to model attentional mechanism. The network activation is fed back and multiplied with input, creating a sigma-pi connection. In sigma-pi connection, the weights are applied to higher level neurons whose activations are computed as the multiplication between the corresponding activations and inputs (Rumelhart and McClelland, 1986). Based on which part of network activation is fed back, we can distinguish the following models:

- A-SRN model (Fig. 2.2) uses output activation to modify network input.
- In case of A-SRN$^+$ model, sigma-pi connection activation from A-SRN is merged with current input, what decrease the propagation of misleading activation in case of incorrect network prediction.
- A-SRN$_{bck}$ model (Fig. 2.3) uses hidden activation with extra set of weights, creating an internal attentional mechanism.

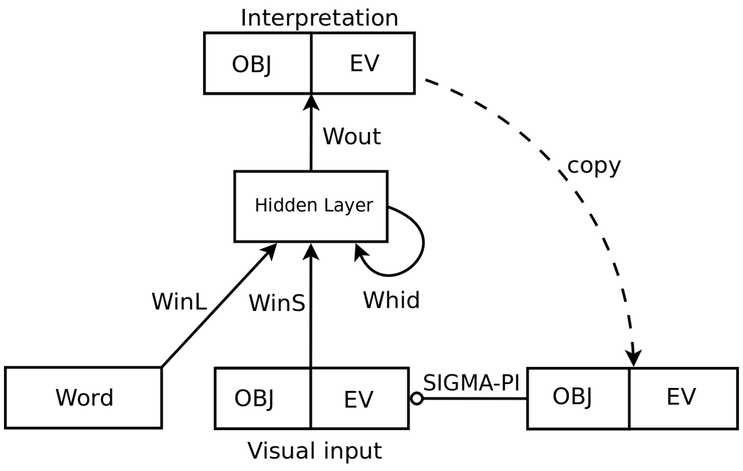


Figure 2.2: Architecture of A-SRN with a language-mediated, top-down attention mechanism. Network output is fed back and multiplied component-wise with the current input via sigma-pi connection.

A-SRN-based models[1] process two types of inputs - the linguistic input $\mathbf{l}_{\text{in}}$ represents spoken language and situational input $\mathbf{s}_{\text{in}}$ describes visual scene via object (OBJ) and event (EV) representations (for further details, see Chapter 6). The A-SRN-based models can also use only one input layer but for the needs of model from Chapter 6 we will describe its more complex form. Sigma-pi connection alter only situational input $\mathbf{s}_{\text{in}}$, leaving linguistic input $\mathbf{l}_{\text{in}}$intact. The activation of the hidden layer for all models (including SRN which is used as reference) is computed as follows:

$$\mathbf{a}_{\text{hid}}(t) = \sigma(\mathbf{W}_{\text{inL}}.\mathbf{l}_{\text{in}}(t) + \mathbf{W}_{\text{inS}}.\mathbf{s}^{'}_{\text{in}}(t) + \mathbf{W}_{\text{hid}}.\mathbf{a}_{\text{hid}}(t-1))$$

where $\mathbf{s}^{'}_{\text{in}}(t)$ expresses:

$$\mathbf{s}^{'}_{\text{in}}(t) = \begin{cases} \mathbf{s}_{\text{in}}(t) & \text{for SRN} \\ \mathbf{s}_{\text{in}}(t). * \mathbf{a}_{\text{out}}(t-1) & \text{for A-SRN} \\ \gamma\ \mathbf{s}_{\text{in}}(t) + (1-\gamma)\ \mathbf{s}_{\text{in}}(t). * \mathbf{a}_{\text{out}}(t-1) & \text{for A-SRN}^+ \\ \mathbf{s}_{\text{in}}(t). * \sigma(\mathbf{W}_{\text{bck}}.\mathbf{a}_{\text{hid}}(t-1)) & \text{for A-SRN}_{bck} \end{cases} \tag{2.1}$$

[1]We will use this term to name all three models A-SRN, A-SRN$^+$ and A-SRN$_{bck}$.

In eq. 2.1 ‘$.*$‘ denotes component-wise multiplication of the two vectors parameter γ influence the effect of input and sigma-pi connection activation. The output activation is similar to one, we could see in SRN:

$$\mathbf{a}_{\text{out}}(t) = [\mathbf{c}_{\text{out}}(t), \mathbf{e}_{\text{out}}(t)] = \sigma(\mathbf{W}_{\text{out}}.\mathbf{a}_{\text{hid}}(t)),$$

where $[u, v]$ stands for concatenation of vectors u and v. For training A-SRN-based models we have used BPTT algorithm. It uses the target $\mathbf{tgt}(t)$ and errors $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3$ for two separate error propagation paths.[2] The following algorithm represents training procedure at time t with window of size $T = 3$. Steps 2 and 3 are repeated T-times (with index $s \in [0, T-1]$) before proceeding to step 4. The symbol $\top$ denotes transpose operation.

```
———————— (0) initialization ————————
$\Delta\mathbf{W}_{\text{inL}}$   :=   $\mathbf{0}$
$\Delta\mathbf{W}_{\text{inS}}$   :=   $\mathbf{0}$
$\Delta\mathbf{W}_{\text{hid}}$   :=   $\mathbf{0}$
$\Delta\mathbf{W}_{\text{bck}}$   :=   $\mathbf{0}$
$\Delta\mathbf{W}_{\text{out}}$   :=   $\mathbf{0}$
———————— (1) output weights ————————
$\mathbf{f}_1$   :=   $\mathbf{tgt}(t) - \mathbf{a}_{\text{out}}(t)$
$\mathbf{f}_1$   :=   $\mathbf{f}_1 .* (\mathbf{a}_{\text{out}}(t))'$
$\Delta\mathbf{W}_{\text{out}}$   :=   $\Delta\mathbf{W}_{\text{out}} + \mathbf{f}_1.\mathbf{a}_{\text{hid}}(t)^\top$
$\mathbf{f}_1$   :=   $\mathbf{W}_{\text{out}}^\top.\mathbf{f}_1$
$\mathbf{f}_2$   :=   $\mathbf{0}$
====== (2) hidden weights ======
$\mathbf{f}_2$   :=   $\mathbf{f}_1 + \mathbf{f}_2$
$\mathbf{f}_2$   :=   $\mathbf{f}_2 .* (\mathbf{a}_{\text{hid}}(t-s))'$
$\Delta\mathbf{W}_{\text{inL}}$   :=   $\Delta\mathbf{W}_{\text{inL}} + \mathbf{f}_2.\mathbf{l}_{\text{in}}(t-s)^\top$
$\Delta\mathbf{W}_{\text{inS}}$   :=   $\Delta\mathbf{W}_{\text{inS}} + \mathbf{f}_2.\mathbf{s}_{\text{in}}'(t-s)^\top$
$\Delta\mathbf{W}_{\text{hid}}$   :=   $\Delta\mathbf{W}_{\text{hid}} + \mathbf{f}_2.\mathbf{a}_{\text{hid}}(t-s-1)^\top$
$\mathbf{f}_3$   :=   $\mathbf{f}_2$
$\mathbf{f}_2$   :=   $\mathbf{W}_{\text{hid}}^\top.\mathbf{f}_2$
$\mathbf{f}_1$   :=   $\mathbf{0}$
```

[2]Just in case of A-SRN-based models; in case of SRN, only one error propagation path is used.

```
————        (3) $feedback$        ————
$\mathbf{f}_1$        :=        $\mathbf{W}_{\mathrm{inS}}^{\top}.\mathbf{f}_3$
$\mathbf{f}_1$        :=        $\mathbf{f}_1.*(\mathbf{a}_{\mathrm{X}}(t-s-1))'.*\mathbf{s}_{\mathrm{in}}(t-s)$
$\Delta\mathbf{W}_{\mathrm{X}}$        :=        $\Delta\mathbf{W}_{\mathrm{X}}+\mathbf{f}_1.\mathbf{a}_{\mathrm{hid}}(t-s-1)^{\top}$
$\mathbf{f}_1$        :=        $\mathbf{W}_{\mathrm{X}}^{\top}.\mathbf{f}_1$
======        (4) $weight\ change$        ======
$\mathbf{W}_{\mathrm{inL}}$        :=        $\mathbf{W}_{\mathrm{inL}}+\alpha\Delta\mathbf{W}_{\mathrm{inL}}$
$\mathbf{W}_{\mathrm{inS}}$        :=        $\mathbf{W}_{\mathrm{inS}}+\alpha\Delta\mathbf{W}_{\mathrm{inS}}$
$\mathbf{W}_{\mathrm{hid}}$        :=        $\mathbf{W}_{\mathrm{hid}}+\alpha\Delta\mathbf{W}_{\mathrm{hid}}$
$\mathbf{W}_{\mathrm{bck}}$        :=        $\mathbf{W}_{\mathrm{bck}}+\alpha\Delta\mathbf{W}_{\mathrm{bck}}$
$\mathbf{W}_{\mathrm{out}}$        :=        $\mathbf{W}_{\mathrm{out}}+\alpha\Delta\mathbf{W}_{\mathrm{out}}$
————        (5) $end$        ————
```

In the above equations, $\mathbf{s}_{\mathrm{in}}^{'}(t-s)$ follows the definition from eq. 2.1 and subscript X in step 3, is notation for output or backward layer, based on type of the model (A-SRN or A-SRN$_{bck}$ respectively). In case of SRN, step 3 is not executed. Model A-SRN$^+$ is trained in the same way as A-SRN.

A-SRN-based models have been used to model explicit utterance-driven visual attention (Švantner et al., 2011b), exhibiting more biologically plausible behaviour compared to a simple recurrent network.

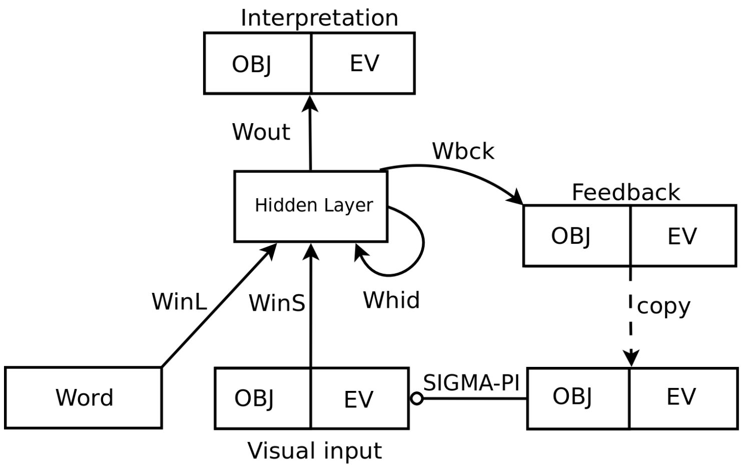


Figure 2.3: Architecture of A-SRN$_{bck}$ model with an internal explicit language-mediated attention mechanism. The hidden layer activation is fed back to input via extra set of weights $\mathbf{W}_{\mathrm{bck}}$.

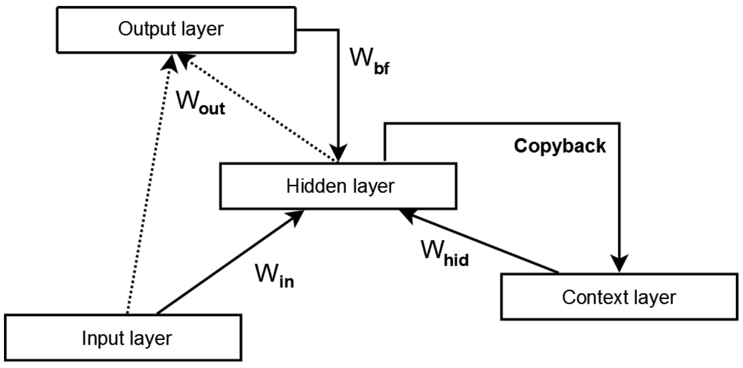


Figure 2.4: Architecture of ESN. Only dashed connections are adjusted during training. Weights in the hidden layer form a dynamical reservoir.

2.3 Echo-state networks

The model introduced by Jaeger (2001) was designed to predict the next input in signal sequences using a more effective training procedure. Echo-state network (ESN) architecture is almost similar to SRN with one crucial difference – recurrent and input weights are not trained (as we can see in Fig. 2.4). Hidden neurons form a dynamical reservoir whose complex dynamics can be easily mapped to the output with a single layer neural network. This allows the usage of linear regression instead of less efficient gradient training methods. Echo-state networks can be extended with optional direct input-output connection and/or feedback connection from output to hidden layer. Optional direct input-output connection is trained. Network activation is expressed as:

$$\begin{aligned}\mathbf{a}_{\mathrm{hid}}(t) &= G(\mathbf{W}_{\mathrm{in}}\mathbf{a}_{\mathrm{in}}(t) + \mathbf{W}_{\mathrm{hid}}\mathbf{a}_{\mathrm{hid}}(t-1) + \mathbf{W}_{\mathrm{bck}}\mathbf{a}_{\mathrm{out}}(t-1))\\ \mathbf{a}_{\mathrm{out}}(t) &= F(\mathbf{W}_{\mathrm{out}}[\mathbf{a}_{\mathrm{in}}(t), \mathbf{a}_{\mathrm{hid}}(t)]),\end{aligned}$$

Hidden activation function $G(x)$ is mostly represented by bipolar sigmoid or *tanh* function, while output activation function $F(x)$ by identity, hyperbolic tangent or sigmoid.

In order for the ESN to work, the reservoir must express contractive dynamics satisfying the echo-state property: *'if given a long enough sequence, the network will always end up in the same state, regardless of the starting state'* (Tong et al., 2007). The resulting state is therefore determined only by the input sequence. Over time, there were discussed several approaches how to achieve the echo-state property. Jaeger (2001) has shown that reservoir which has spectral radius (the largest absolute eigenvalue) of the weight matrix $\lambda(\mathbf{W}) < 1$, leads in most cases to echo-state property. To achieve this, the components $w_{i,j}$ of weights from matrix $\mathbf{W}$ are adjusted according to the formula

$$w_{i,j} = \lambda_{\mathrm{D}} \frac{w'_{i,j}}{|\lambda_m(\mathbf{W}')|},$$

where $\lambda(\mathbf{W}')$ is the largest eigenvalue of the original matrix and λ_{D} is the desired spectral radius. However, it was shown that $\lambda(\mathbf{W}) < 1$ is not a necessary nor a sufficient condition of echo-state property. What we can only claim is, that echo-state property is violated for zero input if $\lambda(\mathbf{W}) > 1$ using reservoir with *tanh* nonlinear neurons (Lukosevicius and Jaeger, 2009). To fulfil an echo-state property, we need to provide the reservoir weight matrix whose greatest singular value meets the condition $\sigma(\mathbf{W}) < 1$ (Jaeger, 2001). Similarly, also maximum singular value of the matrix can be altered to the desired value by modifying the components $w_{i,j}$ of weights from matrix $\mathbf{W}$

$$w_{i,j} = \sigma_{\mathrm{D}} \frac{w'_{i,j}}{\sigma_m(\mathbf{W}')}.$$

Before training, both $\mathbf{W}_{\mathrm{in}}$ and $\mathbf{W}_{\mathrm{hid}}$ weights are usually initialized as sparse random matrices. Only output weights $\mathbf{W}_{\mathrm{out}}$ are adapted during ESN training. Apart from gradient training methods, we can use linear regression, by solving the equation:

$$\begin{aligned} \mathbf{A}_{\mathrm{tgt}} &= \mathbf{W}_{\mathrm{out}}\mathbf{A}_{\mathrm{hid}} \\ \mathbf{W}_{\mathrm{out}} &= \mathbf{A}_{\mathrm{tgt}}\mathbf{A}_{\mathrm{hid}}^{-1}, \end{aligned}$$

where $\mathbf{A}_{\mathrm{hid}} \in \Re^{T \mathrm{x} N}$ is a collected matrix of reservoir activations and $\mathbf{A}_{\mathrm{tgt}} \in \Re^{T \mathrm{x} N}$ is a matrix of target activations (considering N as the number of neurons in reservoir and T as the data set size). Since $\mathbf{A}_{\mathrm{hid}}$ is not a square matrix we must compute its pseudoinverse $\mathbf{A}_{\mathrm{hid}}^{+}$ instead of inverse $\mathbf{A}_{\mathrm{hid}}^{-1}$. Pseudoinverse computation has high numerical stability but is memory demanding for larger data sets. To resolve this problem we can alter previous equations as follows:

$$\begin{aligned} \mathbf{A}_{\mathrm{tgt}}\mathbf{A}_{\mathrm{hid}}^{\top} &= \mathbf{W}_{\mathrm{out}}\mathbf{A}_{\mathrm{hid}}\mathbf{A}_{\mathrm{hid}}^{\top} \\ \mathbf{W}_{\mathrm{out}} &= \mathbf{A}_{\mathrm{tgt}}\mathbf{A}_{\mathrm{hid}}^{\top}(\mathbf{A}_{\mathrm{hid}}\mathbf{A}_{\mathrm{hid}}^{\top})^{-1} \end{aligned} \tag{2.2}$$

The use of eq. (2.2) decreases memory usage because the matrix $\mathbf{A}_{\mathrm{hid}}\mathbf{A}_{\mathrm{hid}}^{\top} \in \Re^{N \mathrm{x} N}$, but the drawback is that numerical stability of the solution is decreased. This can be corrected by Tikhonov regularization (Lukosevicius and Jaeger, 2009).

$$\mathbf{W}_{\mathrm{out}} = \mathbf{A}_{\mathrm{tgt}}\mathbf{A}_{\mathrm{hid}}^{\top}(\mathbf{A}_{\mathrm{hid}}\mathbf{A}_{\mathrm{hid}}^{\top} + \alpha^2 I)^{-1},$$

where $\alpha = 0.5$ is the regularization factor and $I \in \Re^{N \mathrm{x} N}$ is the identity matrix. Matrices $\mathbf{A}_{\mathrm{tgt}}\mathbf{A}_{\mathrm{hid}}^{\top}$ and $\mathbf{A}_{\mathrm{hid}}\mathbf{A}_{\mathrm{hid}}^{\top}$ can be rewritten as:

$$\begin{aligned} \mathbf{A}_{\mathrm{tgt}}\mathbf{A}_{\mathrm{hid}}^{\top} &= \sum_{t=1}^{T} \mathbf{a}_{\mathrm{tgt}}(t)\mathbf{a}_{\mathrm{hid}}^{\top}(t) \\ \mathbf{A}_{\mathrm{hid}}\mathbf{A}_{\mathrm{hid}}^{\top} &= \sum_{t=1}^{T} \mathbf{a}_{\mathrm{hid}}(t)\mathbf{a}_{\mathrm{hid}}^{\top}(t), \end{aligned}$$

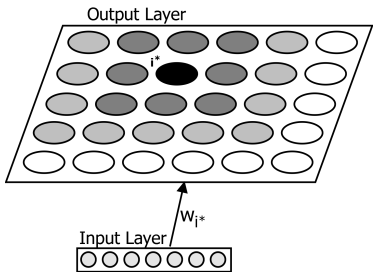


Figure 2.5: Architecture of self-organizing map. Similar inputs are mapped to neighbouring output neurons.

where $\mathbf{a}_{\mathrm{tgt}}(t) \in \Re^{N\mathrm{x}1}$ and $\mathbf{a}_{\mathrm{hid}}(t) \in \Re^{N\mathrm{x}1}$ are target and reservoir activations in time t, respectively. These equations remove memory requirements and allow processing of large data sets. Additionally, ESN can be tested during training to get preliminary results.

Echo-state networks are very popular in prediction of the next input from the symbol sequences (for example, Čerňanský and Makula, 2007). Recently, they were applied also to language domain (Frank, 2006a,b; Tong et al., 2007)). It was shown that ESN has the ability to predict upcoming words, to adapt to long distance dependencies and to facilitate acquisition of grammar structure.

2.4 Self-organizing map

Self-organizing map (SOM) shown in Fig. 2.5 was introduced by Kohonen (1990). Each input activates certain area of network's output what results in topological organization of input vectors (similar objects are grouped together in the resulting map). To achieve this behaviour, the model uses unsupervised learning, selecting winner for each input. After activation, winner's weights and weights of its surrounding neurons are adapted, while other weights are left intact. To find the index i^* of the winner we use the formula:

$$i^* = \arg\ \min_i \|\mathbf{x}(t) - \mathbf{w}_i\|, \tag{2.3}$$

where $\mathbf{x}(t)$ is current input, $\mathbf{w}_i$ is the weight vector for i-th neuron and $\|.\|$ denotes the Euclidean norm. After selection of the winner, surrounding weights are changed using the formula:

$$\triangle \mathbf{w}_i = \alpha(t) h(i, i^*)(\mathbf{x}(t) - \mathbf{w}_i),$$

where $\alpha(t)$ expresses learning rate at time t. Neighborhood function $h(i, i^*)$ between i-th neuron and the winner, can be expressed by two possible distances:

$$h(i, i^*) = \begin{cases} e^{-d(i,i^*)^2/\sigma^2(t)} & \text{Gaussian distance} \\ 1 \text{ iff } d_{\mathrm{M}}(i, i^*) \leq \lambda(t), 0 \text{ otherwise} & \text{Manhattan distance,} \end{cases}$$

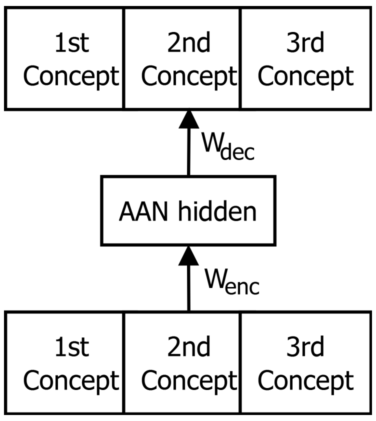


Figure 2.6: Architecture of auto-associative network. This particular network encodes agent-action-patient triplet into one 64 dimensional vector using input matrix $\mathbf{W}_{\text{enc}}$. AAN inner representation can be decoded using output weight matrix $\mathbf{W}_{\text{dec}}$

where $d(i, i^*)$ denotes Euclidean distance and $d_{\text{M}}(i, i^*)$ denotes Manhattan distance between positions of i-th neuron and the winner in the two-dimensional rectangular map. Size of affected neighborhood is let to decrease during training. In case of the Gaussian distance this is done by function $\sigma(t)$ and in case of Manhattan distance by decreasing function $\lambda(t)$. Learning rate is also time-dependent, mostly expressed by functions $\alpha(t) = 1/t$ or $\alpha(t) = \exp\{-k.t\}$ for constant k.

Self-organizing maps are suitable for visualization of high-dimensional data , using their topological mapping property. They can be used also for preprocessing of complex data, where topological organization eases the later data processing (Švantner et al., 2011a). Additionally, more advanced SOM architectures (see Section 2.6) are also used for processing and visualization of data sequences.

2.5 Auto-associative network

Auto-associative network (AAN) (Cottrell et al., 1989) is a feed-forward network with one hidden layer. During training, every input is presented also as a target, what forces network to create its compressed representation on the hidden layer. Auto-associative networks are therefore often used to reduce dimension of object representation (Cottrell et al., 1989). Original representation can be afterwards easily decoded with the AAN output layer. Activation of AAN is expressed as:

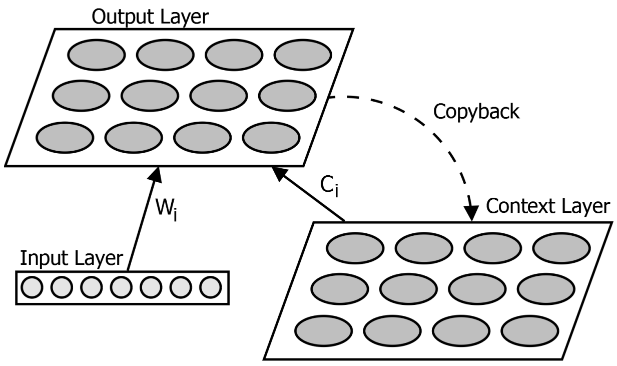


Figure 2.7: Architecture of RecSOM network. Activation of hidden layer is copied to the context layer and presented to hidden layer in next step.

$$\begin{aligned}\mathbf{a}_{\text{hid}}(t) &= G(\mathbf{W}_{\text{enc}}\mathbf{a}_{\text{in}}(t)) \\ \mathbf{a}_{\text{out}}(t) &= F(\mathbf{W}_{\text{dec}}\mathbf{a}_{\text{hid}}(t))\end{aligned}$$

Auto-associative network is often trained with back-propagation training algorithm. Similar inputs tend to have similar AAN codes what leads to resembling behaviour as we could see in self-organizing networks. Use of AAN can be found for example in Švantner et al. (2011b), where we trained the network to create representation of visual event, which respects order of the event constituents. Introduction of the recurrent layer helps auto-associative network to process sequential or structured data. Recurrent auto-associative network (RAAM) was successfully used for this task in Farkaš and Pokorný (2009).

2.6 Recursive self-organizing map

RecSOM architecture was presented by Voegtlin (2002) as an extension of SOM. In each time step, the activation of the output layer is copied to the context layer what allows model to remember previous inputs. Activation of i-th neuron within RecSOM network can be expressed as:

$$y_i(t) = \exp\{-d_i(t)\},$$

where

$$d_i(t) = \alpha\|\mathbf{x}(t) - \mathbf{w}_i\|^2 + \beta\|\mathbf{y}(t-1) - \mathbf{c}_i\|^2$$

Parameters $\alpha > 0$ and $\beta > 0$ influence effect of input and context respectively. Context layer is created in each time step as direct copy of the output layer. After selection of

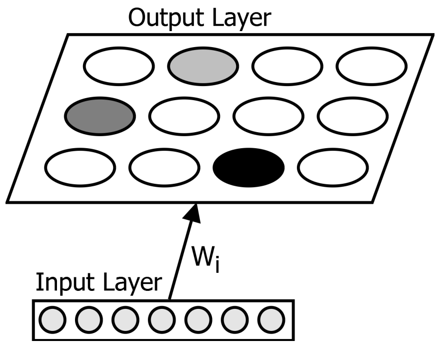


Figure 2.8: Architecture of SardNet network. Previous winners remain activated also after presenting next input. Their activation is decayed by decay factor κ.

the winner, the weights $\mathbf{w}_i$ are adjusted similarly as in SOM and context weights $\mathbf{c}_i$ are adjusted using formula:

$$\Delta\mathbf{c}_i = \eta(t)h(i,i^*)(\mathbf{y}(t-1) - \mathbf{c}_i)$$

RecSOM is often used to represent symbol sequences. Resulting representations are organized in Markovian manner, mapping subsequences with common suffix close to each other (Tiňo et al., 2006). Vančo and Farkaš (2010) compared RecSOM with other recursive SOM models (SOM SD, MSOM) on more complex tree data structures.

2.7 SardNet model

Sequential Activation Retention and Decay Network (SardNet) was introduced by James and Miikkulainen (1995). It originates from SOM architecture, leaving previous winners activated also during succeeding inputs. However, previous winners activation is decayed via formula $y_i := \kappa y_i$ using decay factor $0 < \kappa < 1$. Moreover, each winner is removed from competition and cannot represent later inputs, allowing its neighbours to represent similar future items of sequence.

SardNet can be used only for finite sequences, whereas all other models with self-organization also for infinite sequences. Another difference is that a sequence in SardNet is represented as a distributed pattern (of winners), while in the other models the sequence representation is localist.

SardNet is suitable for representation of symbol and word sequences because gradually activated neurons are able to memorize all items of sequence. Additionally, information about order of presented items is retained too. SardNet was used by Mayberry and Miikkulainen (2003) and Farkaš and Crocker (2006) in memorizing of word sequences task. Farkaš and Crocker (2006) used the combination of RecSOM and SardNet named RecSOMsard

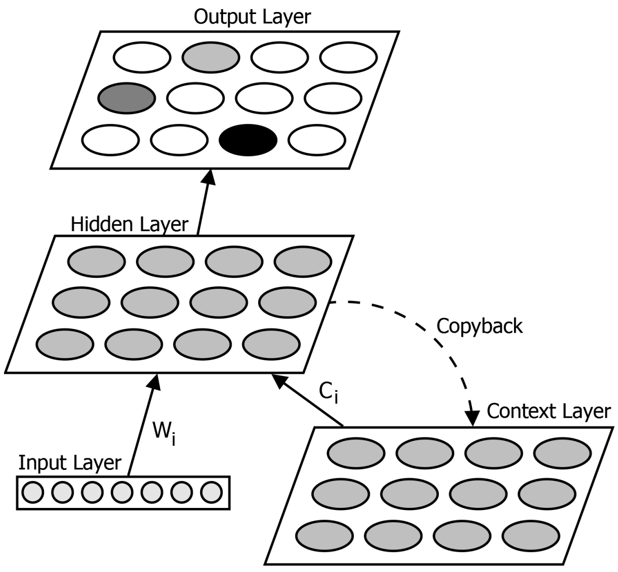


Figure 2.9: The architecture of RecSOMsard network. RecSOMsard is a combination of RecSOM and SardNet model.

for word prediction task. In RecSOMsard, output of RecSOM is transformed to the output with mechanism used in SardNet. The model was able to effectively predict words, proving that models with an unsupervised module are applicable in this domain.

Chapter 3

Prediction paradigm in natural language processing

We can split the natural language processing domain into two main parts. Lexical processing is responsible for word reading, morphology and pronunciation. The higher-level language processing operates on the words and it participates in sentence parsing, comprehension, word prediction and production. Next chapters focus on the prediction task, because we believe that it eases the parsing and comprehension and enables the language production.

3.1 Syllable prediction

The first attempt to model the syllable and word prediction was made by Elman (1990). He trained the simple recurrent network (SRN) to predict the next characters of a simple artificial language. Input of the network was represented by a sequence of concatenated English words (without space between them). The network showed the ability to detect word boundaries at locations of high entropy, demonstrating that word segmentation is facilitated by prediction mechanism. Moreover, according to Mintz et al. (2002) and Saffran (2002), the statistical learning plays an important role in syntax and language morphology acquisition.

A more complex subtask in language processing is the utilization of non-adjacent dependencies between syllables. The non-adjacent dependencies are often used in the natural language, for example representing the common number or tense of the subject-action pair (e.g. in the sentence *Peter <u>has</u> arriv<u>ed</u>*). Peña et al. (2002) experimented with persons and came to the conclusion that people are not able to generalize[1], using only statistical

[1]To use known dependencies with the novel paddings. In the sentence *Peter <u>has</u> arriv<u>ed</u>*, the padding is expressed by syllables '*arriv*'.

information from the language and must utilize additional cues (like pauses between words and phonological characteristics) via algebraic computational (i.e. rule-like) mechanism. Gómez (2002) reproduced these experiments and showed that inability of generalization was caused by characteristics of the used language. Language used in Peña et al. (2002) exhibited only small variability of the padding, what caused that people could rely on adjacent dependencies. The variability of the padding is defined as the number of possible subsequences between the dependent words. Peña et al. (2002) have summarized these findings as the variability hypothesis. Onnis et al. (2003) and Onnis et al. (2004) have further extended the variability hypothesis, stating that word generalization occurs when variability of padding is zero or large. They showed on the input data used in Peña et al. (2002), that people are able to simultaneously segmentate the words and generalize using novel paddings. These findings prove that statistical properties of the input language are sufficient to generalize under certain circumstances.

The segmentation of words and the variability hypothesis was successfully modelled using simple recurrent network by Černák (2005) and Farkaš (2009). Černák (2005) has additionally modelled both tasks using ESN, being successful only in the case of word segmentation. In case of variability hypothesis his model was unsuccessful because the reservoir, as it turned out, was not carefully chosen and initialized.

3.2 Next word prediction

Recurrent neural networks have proven to be a successful modelling tool for natural language processing also on the sentence level. Elman (1991) extended his work and trained an SRN in the next word prediction task. Since then, the SRN has been often used for processing language data (e.g. Lawrence et al., 2000; Christiansen and Chater, 1999a). Elman (1991) demonstrated, that the SRN is able to learn an underlying grammar, when trained on (simplified) English sentences. Moreover, SRN was able to create meaningful context-dependent representations of the words at its hidden layer, despite the fact that it was trained on localist inputs.

Simple recurrent network was able to process complex sentences only after gradual increasing of complexity of sentences during training. This result corresponded to the 'less-is-more' hypothesis introduced by Newport (1990). According to this hypothesis, children's ability to learn the first language to a greater degree of fluency, is caused by infant's limited cognitive abilities. Elman (1993) modelled this hypothesis and showed that memory span impairments at the early stages of child's life and its gradual development also facilitates development of primary language. However, Rohde and Plaut (2003) argued against this hypothesis by claiming that gradual growth of the sentence complexity and memory span impairments are not advantageous but in fact harmful for language acquisition.

Elman's results were repeated by Tong et al. (2007) who additionally compared SRN with an echo-state network. His findings show that the ESN exhibits similar performance as the simple recurrent network. However, it has been argued that the SRN was trained using BPTT training method which does not guarantee best possible results. SRN trained for this task with extended Kalman filter exhibit superior performance, while the performance of ESN is only similar to performance of common statistical methods (like variable-length Markov models).

Čerňanský et al. (2007) showed that activations of recurrent units lead to structural differentiation even before training (architectural bias). Amount of structural differentiation of untrained RNN is comparable to variable length Markov models. They claimed, that fixed-point attractors for each input exist even before training and that they are randomly placed in the state space. During training, attractors change their positions to express characteristics of input data. Attractors for similar data are placed to the same regions.

Farkaš and Crocker (2006) used an alternative approach to modelling the word prediction task. Instead of using supervised learning algorithms, they used a combination of recursive self-organizing map (RecSOM; Voegtlin, 2002) and SardNet (James and Miikkulainen, 1995). They argued, that self-organizing models are more biological plausible and can be easily extended also for more complex languages. After development of inner self-organized representations, these were mapped to predicted words via supervised mechanism. The model had performed in the simulating word prediction task and displayed high robustness after lesioning the hidden layer.

Chapter 4

Syllable prediction

In this chapter we will examine the effect of adjacent and non-adjacent dependencies on the syllable prediction task. We will model the variability hypothesis (Peña et al., 2002) using echo-state network trained on a simple artificial language processed at syllable level. Results of this chapter come from Farkaš and Švantner (2007).

4.1 Experiments

As input data, we will use artificial language concatenated from three syllable words. In total, we will use 3 different input sets S1, S2, S3 with different size of lexicon. The input words are created from 3 different word frames $A_iX_jB_i$: *ba_te*, *gu_do* and *pi_ra*, which are used with same probability. Input sets S1, S2, S3 are created as all possible combinations between word frames and padding sets X1, X2, X3 which we can see in Table 4.1. Syllables from the word frames are never used in the padding. According to Peña et al. (2002), we can divide the words into 3 different groups. The *grammatical words* (GW) are valid words from input sets S1, S2, S3 with the transition probabilities[1] $P(X_j|A_i) = 1/|X|$, $P(B_i|A_i) = 0$, $P(B_j|X_j) = 0.33$ and $P(A_i|B_i) = 0.5$ (because words frames were not allowed to repeat consecutively). Since the input is the sequence of concatenated input words the model may incorrectly detect word boundaries. In this case the *part words* (PW) are created. More formally, the PW is composed from the last (third) syllable of one word and first two syllables of second word or from last two syllables of first word and first syllable of the second word. For example in input:

$$\text{gudido}\overline{\text{badi}\textbf{te}}\textbf{pidi}\text{ra}\overline{\text{gu}\textbf{dido}}\textbf{pi}\text{dira}\overline{\text{badite}}\text{pidira},$$

are some PWs marked with bold and every other grammatical word is overlined. Since the PWs represent alternative possibilities of word division, they can be used for the evaluation

[1] $P(A|B)$ stands for the conditional probability, that event A occurs given that B has occurred.

Set	Content
X1	di
X2	di, ku, to, pa
X3	be, bi, bo, bu, by, ta, ti, to, tu, ty, ga, ge, gi, go, gy, da, de, di, dy, pa, pe, po, pu, py, re, ri, ro, ru, ry

Table 4.1: Padding sets for input languages S1,S2,S3. S1 set represents padding set with no variability (1 syllable), S2 contains four syllables and S3 represent padding set with high variability (29 syllables).

of word segmentation. If the model prefers the GWs over PWs it is able to correctly segment words from the input sequence.

Rule words (RW) are words with schema $A_iA_jB_i$ or $A_iB_jB_i$ (e.g. *bagute* or *pidora* from the $S1 = \{badite, gudido, pidira\}$). Rule words contain syllables from the word frame also in the padding, so we can use them as a generalization of GWs. They have used transition probabilities $P(A_j|A_i) = P(B_i|A_j) = 0$ and $P(B_j|A_i) = P(B_i|B_j) = 0$, with the only dependency between non-adjacent syllables. Rule words have never been used during training and they contain only non-adjacent dependencies in opposition to the part words, so they are suitable as generalization test for matching the non-adjacent dependencies. More formally, if the model counts the adjanced dependencies it should prefer PW over RW in case of small variability of the padding. In case of counting non-adjacent dependencies it should rely on most statistically reliable dependencies, which are in this case $P(B_i|A_i)$. This will allow to create correct segmentation of the words (both GWs and RWs).

4.2 Model specification

The task was modelled using ESN. Its task was to predict next syllable within the continuous sequence. The syllables were encoded with one-hot encoding, creating 7, 10 and 35-dimensional input and output vectors for S1, S2, S3 respectively. The input layer weights of the ESN were set to -0.1 and 0.1 with same probability and reservoir was randomly initialized to contain 50 neurons and have spectral radius $\lambda_{max} = 0.8$. Its sparsity was 80%. The output layer weights were randomly initialized from interval [-0.5,0.5] and altered with the standard delta rule with learning rate $\alpha = 0.1$. As an activation function for reservoir neurons we have used unipolar sigmoidal function and for output neurons we have used softmax function. To evaluate the network preference between two words we have used following procedure:

- we choose N words and N non-words for N=20

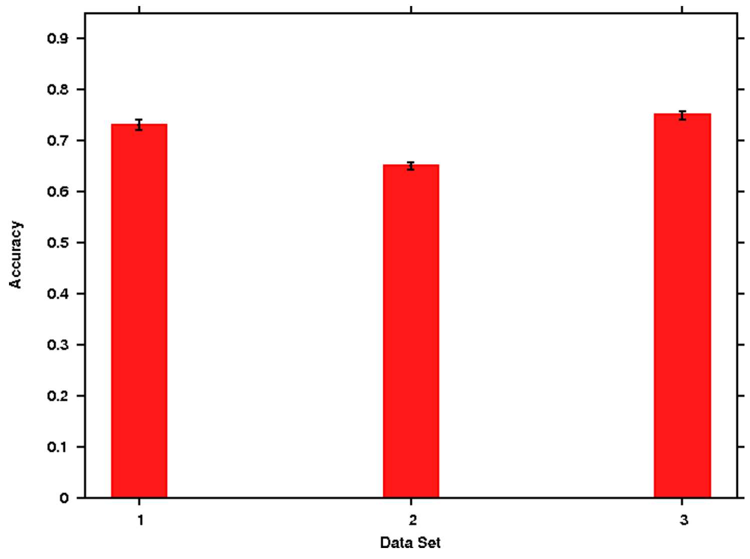


Figure 4.1: Segmentation performance of grammatical words for three input sets with different syllable padding variability.

- for each word, we compute its value: after resetting the reservoir state (to isolate input from previous activations) the first two syllables are presented to the input. Value of the word u is defined as the sum of activations of output neurons corresponding to the next syllables which will be presented at input in second and third step: $u = u(2) + u(3)$.
- we choose N words with largest activation and find how many of them belong to word category (GW or RW) and how many belong to non-word category (PW).

4.3 Results

To test segmentation performance of the ESN model, we evaluated network preference between grammatical GW and part words PW. The model was able to segmentate words successfully for all three input sets as we can see from the Figure 4.1. The average result for S1 was 73%, for S2 65% and for S3 75%.

To test the preference of non-adjacent dependencies we evaluated the model with rule and part words. In S1 the preference in favor of the rule words was 5.5%, in S2 42.5% and in S3 66%. In contrary to the results presented in the Černák (2005), that modelled this task using only small reservoir, we have confirmed the hypothesis of variability for the large padding sets.

Our results does not fully confirm the hypothesis of variability as was predicted by Peña

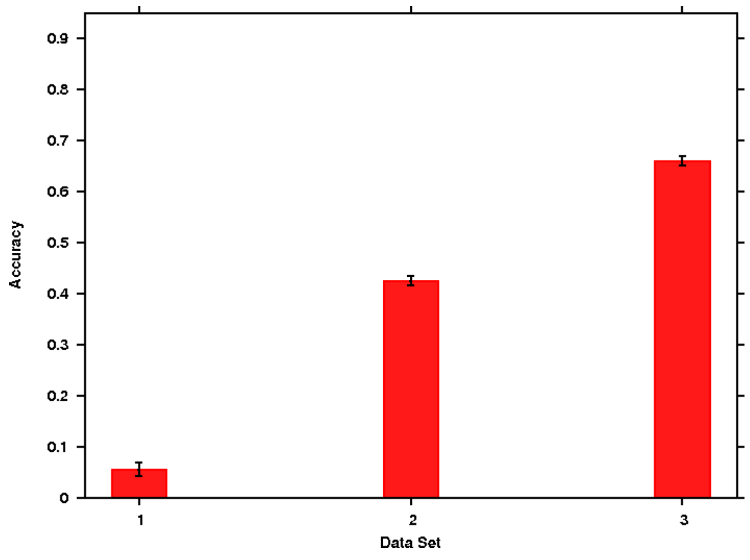


Figure 4.2: Segmentation performance of rule words for three input sets with different syllable padding variability.

et al. (2002) for zero variability. In this case, the preference of rule words is too small. However the preference of RWs for zero variability was shown in the Onnis et al. (2003, 2004) just in case of segmentated input, what simplifies the task significantly.

4.4 Discussion

Learning of the dependencies between elements of sequences occurs rather unconsciously in the form of implicit learning (Cleeremans et al., 1998). Earlier research had pointed out that the people are computing only adjacent dependencies. However, the more recent research shows that also non-adjacent dependencies are used. This method is more combinatorically complex, growing exponentially with the size of the padding. Therefore we can assume that the usage of non-adjacent dependencies is possible only under certain circumstances. It was assumed that these circumstances are valid only in presence of certain additional markings in the text (for example pauses and word segmentation). However, Onnis et al. (2004) has argued that non-adjacent dependencies are used also in continuous sequential input if the variability of the padding is large enough. Our experiments has confirmed this hypothesis for the recurrent neural networks which processes single syllable at time and use the statistical properties of the input text stream.

Chapter 5

Next word prediction

We have focused our work on two areas of natural language processing on sentence level - modelling of grammar acquisition based solely on language input and modelling of utterance-driven attentional mechanism. This chapter is dedicated to grammar acquisition. We will model next word prediction task using a simplified English language. Results of this chapter come from Švantner and Farkaš (2009b,a) and Švantner (2010).

5.1 Experiments

5.1.1 Elman's language

The input data were generated by a stochastic context-free grammar, very similar to one used in Elman (1991) and Tong et al. (2007). The grammar uses a lexicon of the $|\mathcal{L}| = 26$ words which consists of 2 proper nouns, 4 nouns, 7 verbs [1] and inputs *'.'*, *'who'* denoting end-of-sentence marker and conjunction, respectively. Input grammar is defined in Table 5.1. The words can be differentiated into 6 possible categories: proper nouns, nouns, special category for end-of-sentence marker and 'who' and three categories for verbs. These comprise verbs with prohibited object, verbs with mandatory object and verbs with optional object.

The grammar generates the context-free language whose sentences often contain distant dependencies between words. For example, in the sentence '$\underline{\textit{Boy}}$ *who hits John* $\underline{\textit{lives}}$*.*' has the most distant dependency padding of 3 words (between *boy* and *lives*). This recursive structure is very complex to process and multiple embeddings absent in the spoken natural languages (Karlsson, 2007).

[1]Both nouns and verbs are presented in singular and plural forms thus creating 8 and 14 inputs, respectively.

S	$\rightarrow$	NP VP .
NP	$\rightarrow$	PropN | N | N RC
VP	$\rightarrow$	V | V NP
RC	$\rightarrow$	who NP V | who VP
PropN	$\rightarrow$	john | mary
N	$\rightarrow$	boy | girl | cat | dog | boys | girls | cats | dogs
V	$\rightarrow$	chase | feed | see | hear | walk | live | hit
		chases | feeds | sees | hears | walks | lives | hits

Constraints:
- verb number agreement between N and corresponding V
- verb in RC $\rightarrow$ who NP V rule must allow a direct object
- verbs fall into 3 categories:
  - require direct object: VP $\rightarrow$ V NP : (hit, feed, chase)
  - optionally allow direct object: VP $\rightarrow$ V | V NP : (see, hear)
  - preclude direct object: VP $\rightarrow$ V : (walk | live)

Table 5.1: Input grammar used for generating the basic language.

5.1.2 Extended language

Previous grammar is able to construct complex recursive sentences, but it uses just only a small word set. We have extended the grammar to use a more realistic lexicon, extending its size four times. To maintain ratios between different word groups, each of them[2] was enlarged by the same factor. The rules of the grammar were similar as in previous language - the grammars were distinguished only by terminal rules and constraints. We can find altered rules of extended grammar in Table 5.2.

For both languages we have generated a corpus of 5000 sentences, which was then randomly split into training and test sets by ratio 90:10. The corpus contained 75% of complex recursive sentences, in which the rule NP$\rightarrow$N RC was applied at least once. The average length of the sentence for both languages was over 8 words, not counting the end-of-sentence marker.

5.2 The model

To model the grammar acquisition, we have used the echo-state network. ESN was trained to predict next symbol of presented word sequence using randomly initialized reservoir with spectral radius of $\lambda_{max} = 0.98$ and sparsity 27%. As an activation function for

[2]Except trivial '.' and 'who' groups.

PropN$_E$	$\rightarrow$	john | mary | kate | steve | anna | alice | bob | martin
N$_E$	$\rightarrow$	boy | girl | cat | dog | man | woman | rabbit | lion | wolf | bird | bear | mouse | king | queen | soldier | doctor | boys | girls | cats | dogs | men | women | rabbits | lions | wolves | birds | bears | mice | kings | queens | soldiers | doctors
V$_E$	$\rightarrow$	hit | hits | chase | feed | see | hear | walk | live | think treat | like | love | admire | pet | read | write | count |run | hate | beat | poke | hug | swim | understand | know | defend | remember | entertain | educate | chases | feeds | sees | hears | walks | lives | thinks | treats | likes | loves | admires | pets | reads | writes | counts | runs | hates | beats | pokes | hugs | swims | understands | knows | defends | remembers | entertains | educates

Modified constraints:
- verbs fall into 3 categories:
    - require direct object: VP $\rightarrow$ V NP :
      (feed, hit, chase, treat, like, love, pet, hate, beat, poke, hug, defend)
    - optionally allow direct object: VP $\rightarrow$ V | V NP :
      (see, hear, admire, understand, know, remember, entertain, educate)
    - preclude direct object: VP $\rightarrow$ V :
      (walk, live, think, read, write, count, run, swim)

Table 5.2: Input grammar used for generating the extended language.

reservoir neurons we have used tanh function. Output neurons used linear activation function. Word inputs and outputs were encoded with localist (one active neuron for each word) or distributed codes. Localist representation gave us 26 and 98 dimensional input and output, for simple and extended languages respectively. The distributed codes formed representations of various lengths. We will describe them in a more detail in further text.

Due to the lexical ambiguity in forthcoming text, more candidate words are consistent with the grammar, used for generating of the sentences. For this reason, a recurrent network which uses localist representations on the output and is trained for the next word prediction task, learns to approximate true posterior probabilities of the following words rather than representation of the next word. We will name the posterior probabilities according to Tong et al. (2007) as ground truth probabilities. They can be computed from the underlying grammar. The processing of ambiguous sentence can be seen in Fig. 5.2.

Usage of distributed word representations for both inputs and targets requires subsequential transformation of the network output to meet the definition of the probability distribution. The weighted average of representations of possible words can be computed as

$$\mathbf{p}_{\mathrm{G}}^{+} = \mathbf{L}.\mathbf{p}_{\mathrm{G}},$$

where $\mathbf{L}$ is a matrix of word representations (in columns) and $\mathbf{p}_{\mathrm{G}}$ are column vectors for ground truth.

Since the network tries to predict a weighted average of candidate word representations $\mathbf{a}_{\mathrm{out}}^{+}$, for the purpose of using evaluation measures defined below, we need to convert the output vectors back, to obtain posterior probabilities $\mathbf{a}'_{\mathrm{out}}$ of all words in the lexicon. This is achieved by the formula

$$\mathbf{a}'_{\mathrm{out}} = \mathbf{L}^{-1}.\mathbf{a}_{\mathrm{out}}^{+},$$

where $\mathbf{L}^{-1}$ is the inverse of $\mathbf{L}$. In case of a rectangular matrix $\mathbf{L}$, we have used the pseudoinverse $\mathbf{L}^{+}$ intead of $\mathbf{L}^{-1}$.

5.3 Performance measures

To evaluate the network performance, we have used two general (NNL and cos) and three linguistically oriented measures (MPR, VAR, AUC) which have been discussed in Tong et al. (2007).

Cosine between ground truth and the network output is defined as

$$\cos(\mathbf{p}_{\mathrm{G}}, \mathbf{a}'_{\mathrm{out}}) = \frac{\mathbf{p}_{\mathrm{G}}^{\top}.\mathbf{a}'_{\mathrm{out}}}{\|\mathbf{p}_{\mathrm{G}}^{\top}\|.\|\mathbf{a}'_{\mathrm{out}}\|}.$$

Acronym	Description
NNL	*Normalized negative log-likelihood* is the standard statistical measure used for symbolic sequences.
COS	*Cosine* between the network output $\mathbf{a}'_{\text{out}}$ and ground truth vectors $\mathbf{p}_{\text{G}}$.
MPR	*The maximum prediction rate* expresses the number of cases when the network's most predicted word is consistent with the grammar.
VAR	*Verb agreement rate* measures the network ability to predict long-distance dependencies.
AUC	*Area under the receiver operator curve (ROC)* uses the assignment of the words into various semantic categories and provides more details about network's predictions.

Table 5.3: Quantitative measures used for evaluating model's performance.

NNL is computed with formula

$$\text{NNL} = -\frac{1}{T}\sum_{t=1}^{T} \log_{|\mathcal{L}|} a'_{\text{out}}(s_t),$$

where T is number of words in the testing set, $|\mathcal{L}|$ is size of the input language and $a'_{\text{out}}(s_t)$ is the network prediction probability corresponding to the next word (symbol) in the test sequence. The optimal prediction (i.e. if the model predicts the ground truth) results in the language entropy and the model tries to reach this value from above. In case of our data set, the estimated entropy equals 0.574. MPR is computed as:

$$\text{MPR} = N_{\text{MPR}}/T,$$

where N_{MPR} is the number of cases when $\mathbf{p}_{\text{G}}(s^*(t)) > 0$, considering $s^*(t)$ as the word corresponding to the most active neuron of the output layer in time t.
Verb agreement is computed as:

$$\text{VAR} = \text{N}_{\text{V}}^{\text{CN}}/\text{N}_{\text{V}},$$

where N_{V} is the number of the all verbs in the corpus and N_{V}^{CN} is the number of cases when the verbs with the correct number are predicted.
AUC is computed as:

$$\text{AUC} = \frac{1}{2}\sum_{\text{x}\in[0,1]} (\text{R}(\text{x}_2) + \text{R}(\text{x}_1)).(\text{x}_2 - \text{x}_1),$$

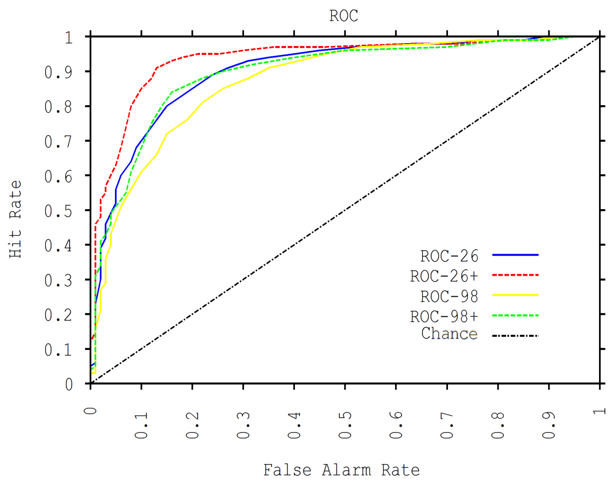


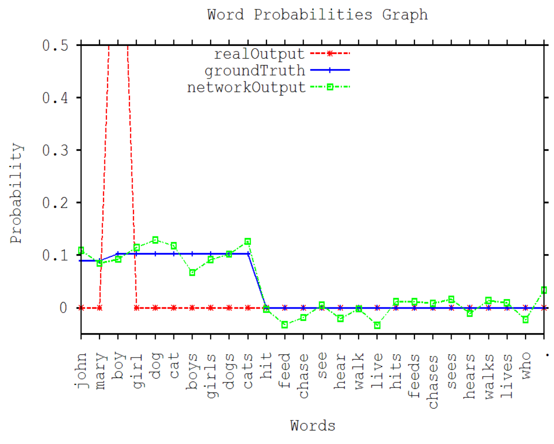
Figure 5.1: ROC curve for different input representations for both simple and extended language.

where $R(x)$ is the value on the ROC curve using x in 0.001 steps. To plot ROC curve we divide the words into 10 semantic categories (nouns, proper nouns, 6 verb groups, *who* and end-of-sentence marker) and threshold the network's output (using a parameter $\theta \in$[0,1] in 0.001 steps) to determine which kinds of words are "possible" and which are "impossible". For this, we compute *hit* and *false alarm* rates for each category. True positives (hits) occur when the mean activation of a feasible class is above θ, while false positives (false alarms) occur when a class is impossible despite being its average activation above θ. Dividing the count of true positives by the number of currently possible classes yields the hit rate, while the false alarm rate is formed by dividing the false positives by the count of currently impossible classes. The ROC is obtained by plotting the hit rate as a function of the false alarm rate. The example of the receiver operator curve curve can be seen in Figure 5.1.

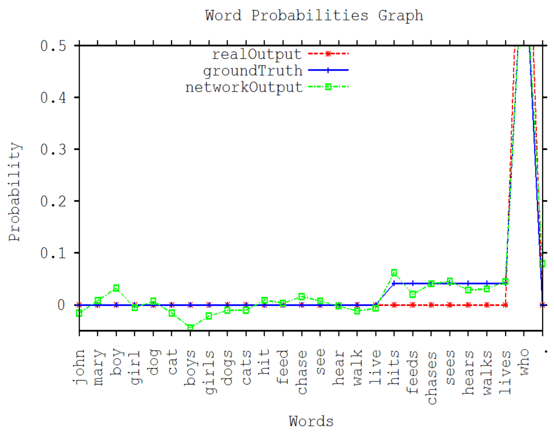
5.4 Basic results

In the fist part we will investigate the performance of ESN on the basic and extended languages. The results show that the model relatively successfully predicts future words of language and is able to acquire properties of an underlying grammar. These findings are supported by both statistical (COS, NNL) and linguistic measures (MPR, VAR, AUC). Model's performance has decreased during processing of the extended language. The complexity of the language caused that ESN was able to assign words into desired categories only to lesser degree of accuracy.

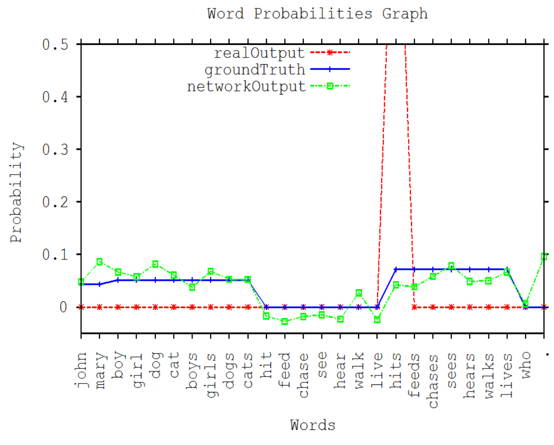
As we can see from Figure 5.3, the input and hidden representations of the ESN do not



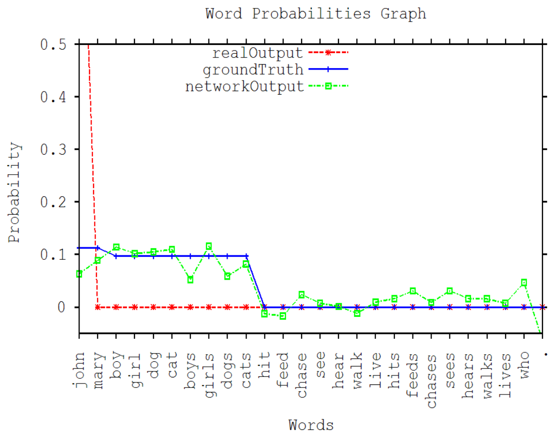
Boy



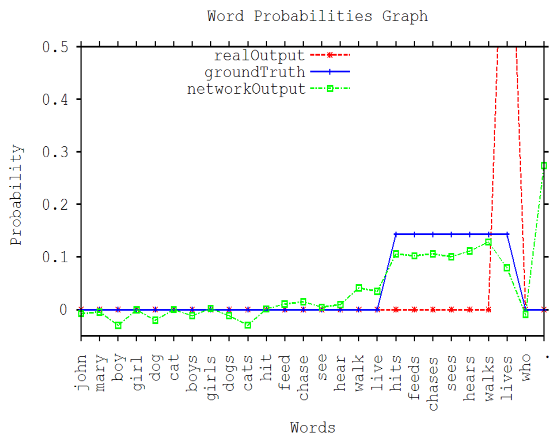
who



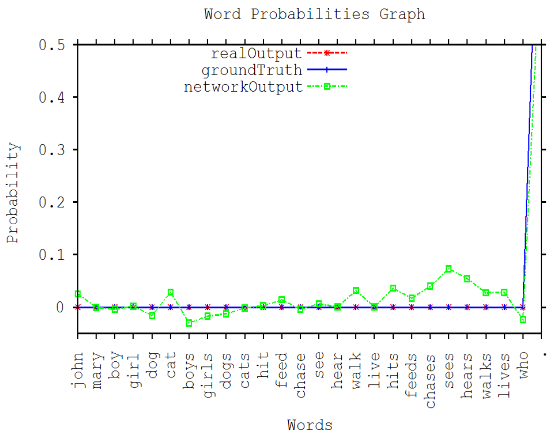
hits



John



lives



.

Figure 5.2: The processing of the sentence "*Boy who hits John lives.*" by the echo-state network. We can notice that the output of the network is consistent with the ground truth.

Model	NNL	COS	MPR	VAR	AUC
ESN 50	0.686	0.831	0.856	0.805	0.874
ESN 300	0.666	0.848	0.884	0.839	0.899
ESN 1000	0.664	0.837	0.897	0.861	0.901

Table 5.4: Performance of ESN for the basic language.

Model	NNL	COS	MPR	VAR	AUC
ESN 50	0.701	0.781	0.782	0.743	0.838
ESN 300	0.687	0.792	0.832	0.816	0.88
ESN 1000	0.699	0.734	0.791	0.833	0.876

Table 5.5: Performance of ESN for the extended language.

carry any semantic information. After the training, the output layer activations separates the words into semantic categories. The activations were obtained similarly as in Elman (1990), by taking the average activation of the particular layer for given word.

5.5 Role of input representations

In the later work, we have focused on the effect of various input representations on ESN performance during language data processing. Apart from localist input and output representations, we have explored three types of distributed word representations: one with random components, ESN$^+$ based and WCD based representations. The latter two are created by corpus preprocessing with statistical methods, while random representations are represented by vectors whose components are uniformly drawn from the interval $[0, 1]$. To investigate the effect of vector dimensionality we have generated random vectors with 26 and 52 components.

ESN$^+$ word representations, introduced in Bullinaria and Levy (2007) and applied in Frank and Čerňanský (2008) are based on word co-occurrences. The j-th component of i-th word is computed as

$$l_{i,j} = N.\frac{N(i,j) + N(j,i)}{N(i).N(j)}, \tag{5.1}$$

where N is the number of all words in the corpus, $N(i,j)$ is the number of times when word with index i immediately precedes word with index j and $N(i)$ is the number of occurrences of the word with index i in the corpus. ESN$^+$ representations (matrix L_{ESN^+})

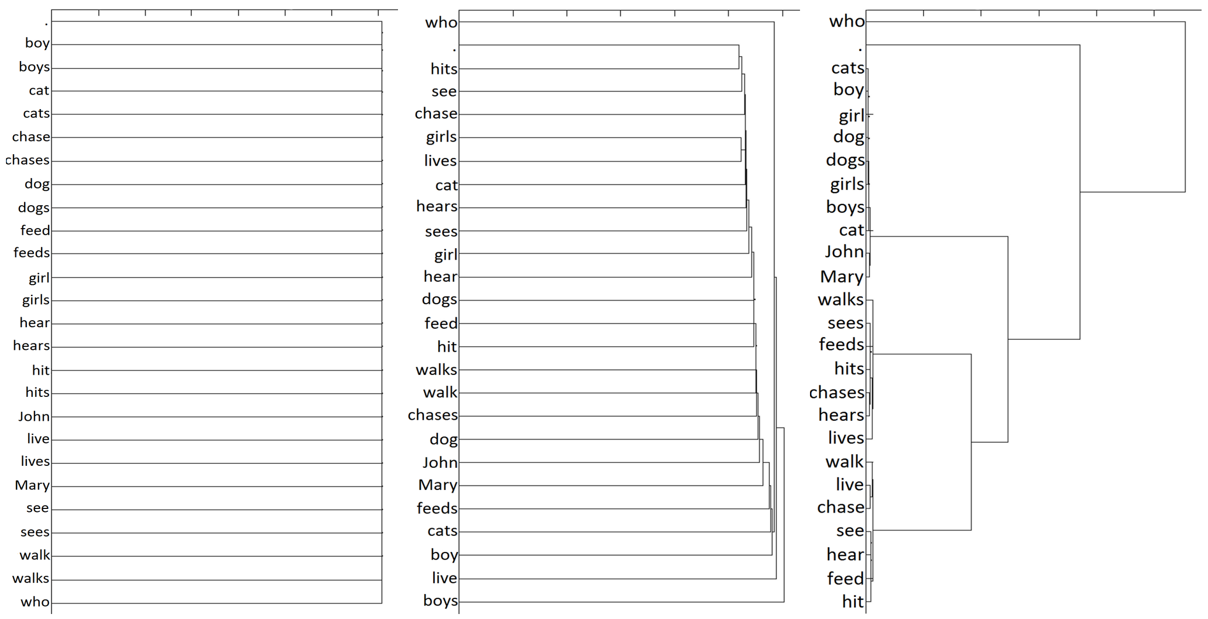


Figure 5.3: Hierarchical cluster diagram of the input, hidden and output representations for ESN model and simple language. Words connected with the lines more to the left are more similar. We can see grouping of words based on their semantic properties.

reflect only word neighbourhoods, ignoring their mutual position. As we can see from its definition, components of this representation are not normalized and can contain values larger than one. To equalize the scale of ESN$^+$ with other examined representations, it was normalized to L_1 norm (city-block distance).

WCD representations (Li et al., 2004) improves ESN$^+$ in two ways. It evaluate left and right co-occurrences separately, hence distinguishing mutual positions of words. Additionally, they evaluate also non-adjacent word co-occurrences, taking into consideration contexts of assumed length X.

$$l^L_{i,j} = \sum_{k=0}^{X-1} \beta^k N_k(i,j) \quad \text{and} \quad l^R_{i,j} = \sum_{k=0}^{X-1} \beta^k N_k(j,i) \tag{5.2}$$

In previous equations, $N_k(v,w)$ is the number of cases when distance of words v and w [3] is k, X is the context width and β is a context-decay parameter. Both $\mathbf{l}^{\mathrm{L}}_i$ and $\mathbf{l}^{\mathrm{R}}_i$ vectors are normalized to L_1 norm to become probabilities. Additionally, also concatenated word representation vectors $[\mathbf{l}^{\mathrm{L}}_i, \mathbf{l}^{\mathrm{R}}_i]$ with 52 components are evaluated, creating the matrix L^{WCD}_{ESN}. To describe different properties of our models, we use the following notation: CLx,CRx,CLRx denote WCD with left, right and both contexts respectively, with context size of x, Ry are ESNs with random vector representations with vector size y.

[3]Number of words between them.

Model	NNL	COS	MPR	VAR	AUC
CL1	0.658	0.841	0.885	0.833	0.887
CR1	0.653	0.836	0.892	0.838	0.891
CLR1	0.651	0.842	0.895	0.832	0.891
EN	0.653	0.835	0.896	0.862	0.900
R26	0.658	0.844	0.878	0.853	0.897
R52	0.653	0.847	0.885	0.858	0.905

Table 5.6: Performance of ESN with distributed representations without scaling.

Model	NNL	COS	MPR	VAR	AUC
CL1x10	0.628	0.880	0.920	0.873	0.918
CR1x10	0.625	0.872	0.918	0.884	0.923
CLR1x10	0.614	0.883	0.919	0.875	0.928
ENx10	0.632	0.867	0.915	0.893	0.921
R26x10	0.662	0.858	0.862	0.833	0.906
R52x10	0.663	0.861	0.858	0.840	0.907

Table 5.7: Performance of ESN with distributed representations scaled by the factor of 10.

For the purpose of studying of the effect of vector scaling, all the representations were scaled up in a component-wise manner to increase the discrimination of words. Motivation for this step was the fact that the input weights of the ESN are not trained and therefore the model cannot use the inputs in the most effective way. Our notation is as follows: xZ means scaling the representation vector by a factor of Z and EN is the normalized version of the ESN$^+$ model (original ESN$^+$ has scaling of order 10).

Table 5.6 shows that differences between normalized ESN$^+$ and WCD representation for left, right and both contexts are very subtle. However, as seen in Table 5.7, the scaling of components improves performance of all models and emphasizes differences between them. With larger scaling, the differences between the model with random representations and semantic representations are magnified – the largest contrast is evident in case of linguistic measures (MPR, VAR, and AUC). These differences are also illustrated in Figures 5.4, 5.5 and 5.6. Somewhat surprisingly, the context size and the size of the representation vector do not play any visible role in model's performance.

As we can see from Figure 5.7, the input and hidden representations of ESN$^+$ model carry slight semantical information, in contrary to ESN representations (see Figure 5.3).

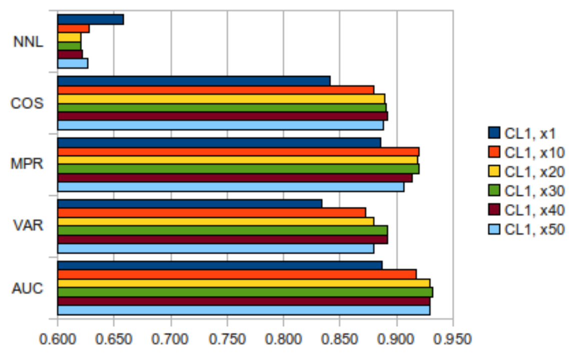


Figure 5.4: Performance of ESN with CL1 word representations with various scales.

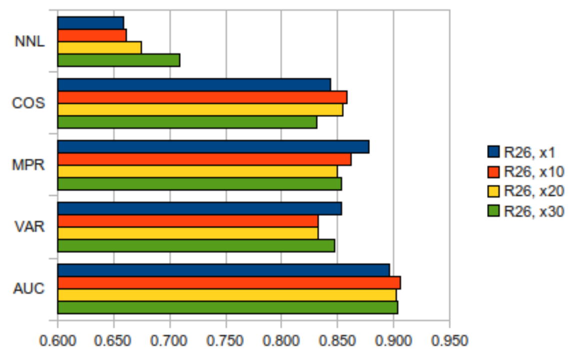


Figure 5.5: Performance of ESN with random word representations with various scales.

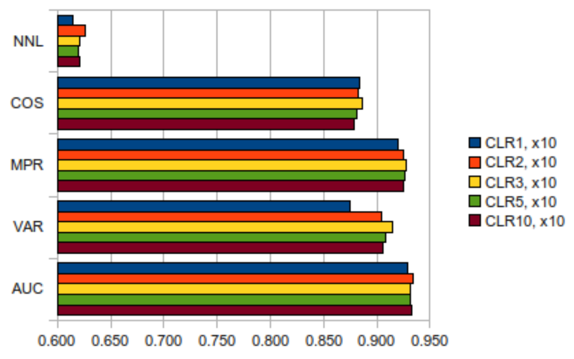


Figure 5.6: The effect of the size of context used to create CLx word representations, on the performance of ESN. Input representations were rescaled with the factor of 10.

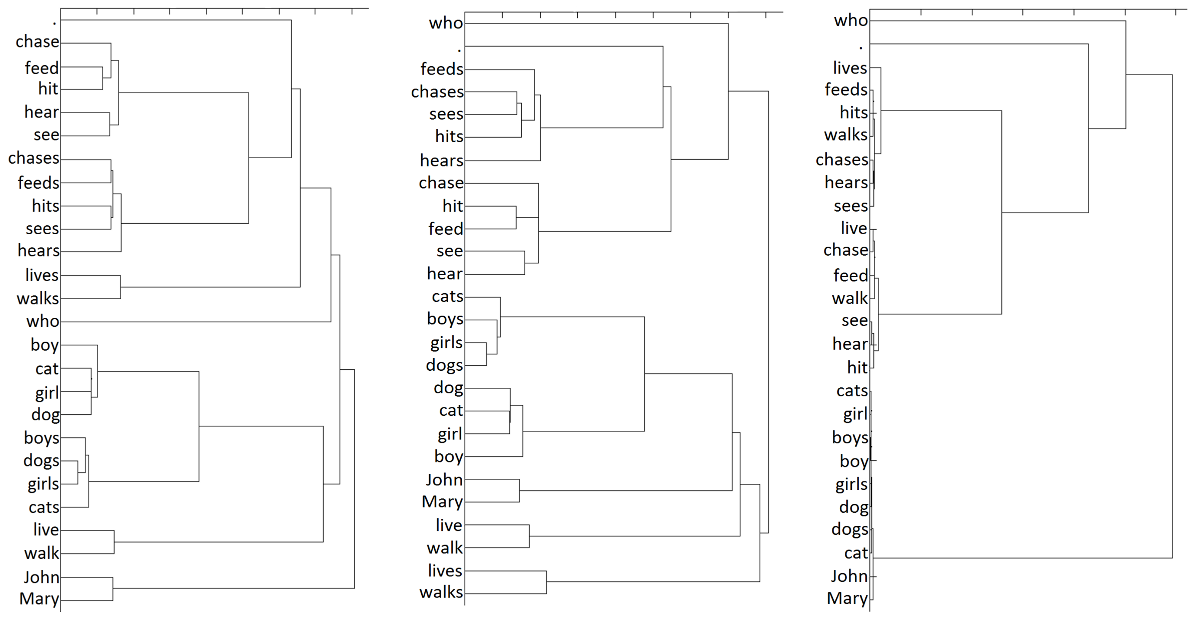


Figure 5.7: Hierarchical cluster diagram of the input, hidden and output representations for ESN$^+$ model and the simple language. Words connected with the lines more to the left are more similar.

Additionally, the output layer representations create better semantical clusters.

5.6 Construction of distributed representations using ESN

The next possible step how to improve the language processing capabilities of the echo-state networks, is to construct distributed input representation using the recurrent neural network. Another instance of the echo-state network, which uses only localist input and output representations, can be used for this purpose. This setup frees us from using the statistical language preprocessing techniques. ESN stores inner representations within its output layer weights $\mathbf{W}_{\text{out}}$ (because only these weights are actually trained). These weights can be directly used as the word representation matrix L_{ESN2} of the ESN2 model. The transposed matrix $L^{\top}_{ESN2}$ can be used similarly as we have used matrices L_{ESN^+} and L^{WCD}_{ESN}.

During the creation of the ESN2 representations, we must take into consideration size of hidden layer during first pass of network. When first instance of ESN had reservoir with n neurons the ESN2 representation will have $n + || + \infty$ components (using ESN implementation which have direct connection between inputs and targets and take into account threshold of the output layer).

Table 5.8 shows that ESN2 representations improved the performance of the ESN and

Model	NNL	COS	MPR	VAR	AUC
ESN	0.652	0.858	0.877	0.864	0.910
ESN$^+$	0.633	0.886	0.89	0.882	0.926
ESN2-28	0.624	0.891	0.901	0.849	0.926
ESN2-327	0.622	0.888	0.903	0.858	0.930

Table 5.8: The performance of various ESN models for simple language.

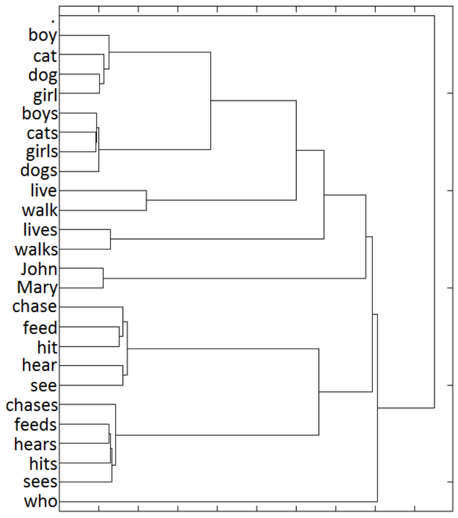


Figure 5.8: Hierarchical cluster diagram of the input for ESN2 model and simple language. Words connected with the lines more to the left are more similar.

were even more successful than ESN$^+$ model. Usage of the larger reservoir in the first pass (300 neurons) had only slight significance over the usage of reservoir with only 1 neuron, showing that the network with very small reservoir is able to provide sufficient word representations. All input representations were rescaled by the factor of 6.0 to magnify the differences between compared models.

Figure 5.8 shows that the input ESN2 representations carry very similar semantical information as we could see in ESN$^+$ model. These were created without an offline statistical method.

5.7 Summary

This chapter was dedicated to the description of grammar acquisition process for simplified context-free language. To model this task we have used the echo-state network.

Although its performance is inferior compared to the SRN[4], but it has very effective training method and therefore allows processing of the larger languages. However, the usage of extended language leads to deteriorated network performance, showing that ESN in its basic implementation suffers from the size of input language. This fact could be caused by usage of the small training corpus (we have used 5000 sentences, for both simple and extended language).

In further work, we examined various types of input and output representations in ESN and their effect on network performance. Apart from localist and random distributed codes, we evaluated language dependent representations, which were developed by the preprocessing of the input corpus. We compared different methods for forming semantic word representations - two methods with statistical preprocessing (ESN^+ and ESN_{WCD}) and one with solely ESN-based representation (ESN2). Networks with various semantic representations showed, somewhat counter-intuitively, very similar performance, ignoring the size of input representation, method used for its creation, the size and type[5] of context during preprocessing.

The most visible effect on the ESN's performance had the scale of the input representations. Network performance was improved, revealing the differences between models. The larger scale of input representations had favored the representations developed by the statistical and ESN-based preprocessing over the symbolic (localist or random codes) ones. This shows that previous results from Frank and Čerňanský (2008) were caused not only by usage of semantic information in word representations but also by the correct scaling of the input vectors (because the ESN^+ representations have, given their definition, a large scale). This fact is the consequence of the ESN's simplified training method - untrained input weights cannot correctly modify contribution of the inputs and therefore the input representations needs to be correctly initialized.

[4]when trained with an extended Kalman filter

[5]the type of context (left, right, both) and presence of the word order.

Chapter 6

Modelling of utterance-driven visual attention

In previous chapters, we have considered language processing as an independent domain. However, human language typically does not occur in isolation. It is connected to surrounding word, most commonly using visual context as source of data. Results of this chapter come from Švantner et al. (2011b), Švantner et al. (2011a) and Švantner (2011).

6.1 Current state of the research field

During the last decade, research into human language comprehension using the *visual world paradigm* has revealed that spoken language can guide attention in a related visual scene. Moreover, scene information can immediately influence comprehension processes (Tanenhaus et al., 1995). Findings have revealed rapid and incremental influence of visual referential context (Spivey et al., 2002; Tanenhaus et al., 1995) and depicted events (Knoeferle et al., 2005) on ambiguity resolution in online-situated utterance processing. Further research demonstrated that listeners even anticipate likely upcoming role fillers in the scene based on their linguistic and general knowledge (Kamide et al., 2003). Knoeferle and Crocker (2006) identified several cognitive characteristics based on the above mentioned findings, claiming that situated language comprehension is incremental, anticipatory, integrative, adaptive, and coordinated, which led to the proposal of the coordinated interplay account (CIA). In more detail, the interpretation of utterance should be developed after each word (language comprehension is incremental), with ability to predict succeeding constituents of target utterance (anticipatory). Moreover, the language comprehension should adopt multiple information sources simultaneously (integrative), exploiting relevant information as soon as it is accessible (adaptive). Moreover, the information sources can depend on each other temporarily (language comprehension is coordinated).

The recent CiaNet model (Mayberry et al., 2009) instantiates the Cia's proposal and accounts for a range of observed empirical findings. CiaNet is a recurrent sigma-pi neural network that models the rapid use of scene information, exploiting an utterance-mediated attentional mechanism. The model was shown to achieve high levels of performance (both with and without scene contexts), while exhibiting hallmark behaviors of situated comprehension, such as incremental processing, anticipation of appropriate role fillers, as well as the immediate use and priority of depicted event information through the coordinated use of utterance-mediated attention to the scene. Other models which link language with the visual world can be found for example in Roy (2005) and Yu et al. (2005).

6.2 Experiments

In the following two chapters, we will investigate a more general network architecture that learns to adapt the explicit attention mechanism. Attention mechanism helps the network to focus on described event and predict its upcoming relevant constituents. Our implementation, in contrary to former models, is able to process more complex scenes[1] and allows inhibition to operate at both the object and event levels (inhibition in CiaNet operates only at event level.).

The network architecture is based on a simple recurrent network (SRN) and recurrent network with explicit attentional mechanism (A-SRN-based models). The networks reconcile an incrementally presented utterance with a representation of the current visual context to predictively recover the described event representation. Language is presented in form of short sentences and the objects and events in the visual world are encoded by scene representations. In each trial, the scene representation is presented at the visual input and the associated sentence is presented at the linguistic input, one word a time. The network's task is to produce the relevant scene representation at the output. This process is mediated by the hidden layer that combines scene-related representations with the symbolic language. The target is available at the output during the entire sentence processing. The explicit feedback (from the output or hidden layer) is added to the network using a sigma-pi mechanism to model the process of focusing attention to relevant constituents shown in the visual scene and mentioned in the associated sentence.

6.2.1 Scene representations

The scene representations are postulated to exist at two levels – the object-level (OBJ) and the event-level (EV). The objects may be the constituents of the events – corresponding to physical agents/patients that can be focused on, whereas the event level refers to specific

[1]The attention mechanism in CiaNet is restricted to favor one of the two concurrent events.

actions in the concrete context (with given semantic roles, i.e. known agent and patient). The combination of both levels of representation is hence assumed to constitute a *semantic representation* of an event. The scene is assumed to consist of multiple events that may or may not share a constituent (e.g. an agent of one event is a patient of the other event or events share common patients), plus random count of distractors (see Fig. 6.1).

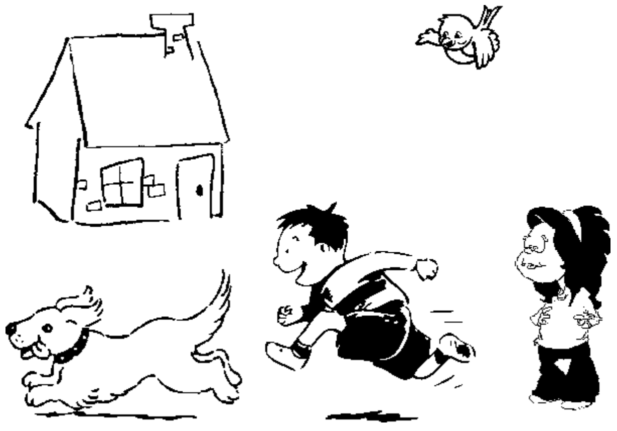


Figure 6.1: Example of a scene consisting of two events *(boy chases dog)* and *(girl looks-at boy)* and two distractors *(house, sparrow).* Both events share the constituent *(boy).*

Objects

Objects include human agents (e.g. *toddler, woman*), animate agents (e.g. *dog, donkey*) and one artificial agent (*robot*) that can be involved in various meaningful activities, with or without a patient. Agents can operate on machines (*forklift, bulldozer*)[2], on objects (e.g. *barrel, house*) or food items (e.g. *apple, juice*). The actions include moving (e.g. *walks, sits*), physical manipulation (e.g. *lifts, holds*), socially oriented activities (e.g. *greets, looks-at*) and sustenance actions (*eats, drinks*). Agents and patients are manually assigned binary features that encode various physical and functional properties and form 40-dimensional vectors $\mathbf{c}_{\mathrm{A}}$ and $\mathbf{c}_{\mathrm{P}}$, respectively. For further details see Tables 6.4 and 6.3. Each object can take a role of a distractor, denoted as $\mathbf{c}_{\mathrm{D}}$, using the same representation vector. Analogically, actions are described by 16-dimensional vectors of binary features $\mathbf{c}_{\mathrm{V}}$. Actually, action encoding consists of only 8 binary features (see Tables 6.1 and 6.2), but the vectors were doubled to increase the differentiation of compressed event representations, performed by EV module.

We used the standard self-organizing map to learn the localized representations of objects. The SOM is constructed in advance, using only agents $\mathbf{c}_{\mathrm{A}}$, patients $\mathbf{c}_{\mathrm{P}}$ and distractors

[2]In fact, these can serve as agents of some actions, too.

	A	B	C	D	E	F	G	H
walk	1	0	1	0	0	0	0	0
run	1	0	1	0	1	0	0	0
sit	1	0	0	0	0	0	0	0
meditate	0	0	0	0	0	0	1	0
lift	1	1	1	1	1	1	0	0
push	1	1	1	1	1	0	0	0
pull	1	1	1	1	1	0	1	0
touch	1	1	0	1	0	1	0	0
hold	1	1	0	1	1	0	0	0
point-at	0	0	0	1	0	0	0	0
look-at	1	0	0	1	0	0	0	0
greet	0	0	0	1	0	1	0	0
hit	0	1	1	1	1	1	0	0
chase	1	0	1	1	1	0	0	0
eat	1	0	0	0	0	1	1	0
drink	1	0	0	0	0	1	1	1

Table 6.1: Feature vectors for actions.

A	animate
B	requires physical contact
C	motion
D	transitive
E	requires physical effort
F	temporary
G	ego-centered (towards oneself)
H	creates flow (liquids)

Table 6.2: Verb feature labels. The labels can be linked with the Table 6.1.

A	small	K	animal	U	vehicle	e	four-legged
B	small-to-medium	L	artefact	V	juicy	f	winged
C	medium	M	food-stuff	W	container	g	wheeled
D	very small	N	male	X	structure	h	causal agent
E	large	O	female	Y	dessert	i	instrument
F	medium-to-large	P	canine	Z	staple	j	manipulate crates
G	very large	Q	feline	a	drink	k	demolish
H	animate	R	bird	b	self-moved	l	store objects
I	inanimate	S	exotic	c	stationary	m	store liquids
J	human	T	machine	d	two-legged	n	storage

Table 6.3: Object feature labels. The labels can be linked with the Table 6.4.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
toddler	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0
boy	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0
girl	0	1	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0
man	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0
woman	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0
dog	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
cat	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
sparrow	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
elephant	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
donkey	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
robot	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0
forklift	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0
bulldozer	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	1	0	0	0
apple	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
crate	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
barrel	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0
wardrobe	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
car	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0
house	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
cake	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
meat	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
milk	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
juice	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
toy	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0

Table 6.4: Feature vectors for objects. Each object has associated 40 dimensional vector of features. Features are represented by *'is present flag'* - if the object exhibit given feature, the relevant dimension of its vector is set to 1, otherwise it is set to 0.

$\mathbf{c}_{\mathrm{D}}$ as inputs, one at the time. Actions are excluded from SOM training, they are included only in event-level representation. The SOM is trained to provide a topographically organized map of objects according to their semantic features. Each object is represented in the SOM by three most active units, focused around the winner, all other units are set to zero. The index of winner i^* was calculated with formula 2.3, having $\mathbf{x}(t) \in \{\mathbf{c}_{\mathrm{A}}, \mathbf{c}_{\mathrm{P}}, \mathbf{c}_{\mathrm{D}}\}$. The activity of the three most active units was rescaled such that $y_{\mathrm{bmu}} = 1$. Resulting map for different objects can be seen in Fig. 6.2. The SOM size was chosen to have 64 units to allow unambiguous learning of each object (by assigning it a separate winner).

The purpose of using three most active units (instead of just a winner) is to allow the activation overlap between similar objects with neighboring winners (this actually helped the model to generalize better. The scene representation at the object level contains the superimposed representations (in SOM) of all objects in the current scene (i.e. all being simultaneously present) plus a few distractors resulting in SOM activation:

$$\mathbf{c}_{\mathrm{in}}^{\mathrm{all}} = \mathbf{c}_{\mathrm{in}}^{(1)} + \ldots + \mathbf{c}_{\mathrm{in}}^{(m)} + \mathbf{c}_{\mathrm{D}}^{(1)} \ldots + \mathbf{c}_{\mathrm{D}}^{(n)},$$

where m and n denote the number of different objects and distractors in the scene, respectively.

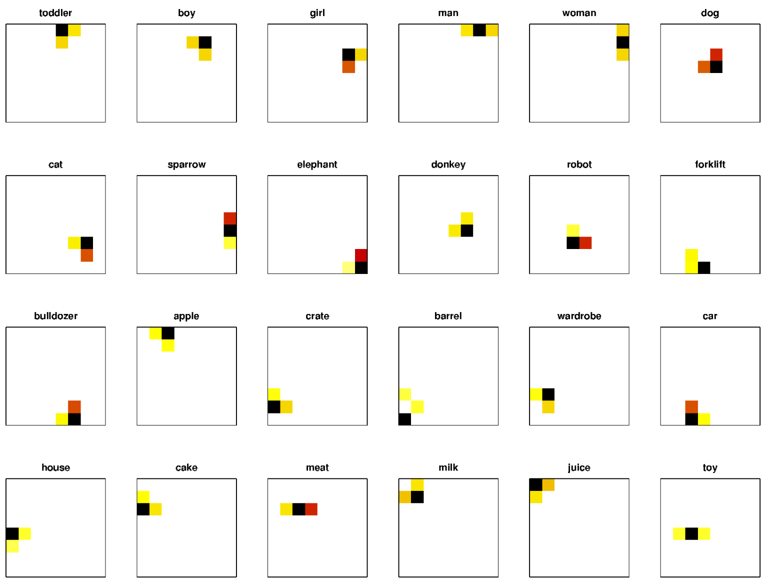


Figure 6.2: SOM representations of all objects used in the simulations. Topographic order according to semantic similarities is evident. Localist nature of representations allows their combinatorial use without causing interference.

Events

To obtain representations $\mathbf{e}_{\text{in}}$ of events, AAN is pretrained off-line on vectors $[\mathbf{c}_{\text{A}}\ \mathbf{c}_{\text{V}}\ \mathbf{c}_{\text{P}}]$ to form the compressed distributed representations at the hidden layer with 48 units. Patient $\mathbf{c}_{\text{P}}$ is optional, so its components are set to zero in case of its absence. The input size dimension for training AAN was 40+16+40=96 dimensions. The functionality of a trained AAN was checked via accuracy of compressed representations using the encoding and decoding of novel agent-action-patient triplets. The accuracy reached almost 100% for testing data.

Once the AAN is trained, the event-level representation corresponding to a scene is taken as a superposition of all events in the situation, resulting in the vector:

$$\mathbf{e}_{\text{in}}^{\text{all}} = \mathbf{e}_{\text{in}}^{(1)} + \ldots + \mathbf{e}_{\text{in}}^{(k)}$$

.

The vector components are constrained in the interval [0,1]. Some components of the event vector could become larger than one after superposition (i.e. if both events had the same unit very active), therefore all components were divided by the value of the most active component. Using the superposition is analogous to that used in CIANET – it encodes simultaneous information provided to the subject as the visual input. However, in CIANET the representational media for two events are separated whereas in our models the medium is shared. Unlike localist representations for objects, the superposition of distributed event representations leads to an overlap between the two codes which expectantly makes the decompression task more difficult. We experimented with decreasing this overlap by manipulating the profile of the sigmoidal activation function (gain and shift) of the hidden units of AAN in order to get sparser compressed codes, but this had no significant effect.

6.2.2 Linguistic representation

The lexicon consists of 40 words, with one-to-one mapping to the objects and actions. Words are treated as symbols and are assigned one-hot codes with 40-dimensions creating an input $\mathbf{l}_{\text{in}}$. The sentences have a SV(O) form, such as *toddler looks-at crate* or *woman walks*.

6.3 Model and network training

6.3.1 Model

To model utterance-mediated visual attention we have used a SRN, echo state network and recurrent networks with attentional mechanism (for more information see chapters 2.1, 2.3 and 2.2). All models use two output slots: $\mathbf{c}_{\text{out}}$ is the object-level output that tries to activate the target objects, taking part in the described event, while $\mathbf{e}_{\text{out}}$ predicts the representation of the target event. Together, the network output (predicted scene interpretation) is given as $\mathbf{a}_{\text{out}} = [\mathbf{c}_{\text{out}}, \mathbf{e}_{\text{out}}]$. The models have no linguistic output.

In total, we have experimented with five models. Beside the standard SRN, ESN and A-SRN we have explored also A-SRN$^+$ model, which was motivated by our initial observations about the effects of feedback mechanism and was designed to help the network avoid undesired object inhibition. A-SRN$^+$ guarantees that input representation remains preserved to a certain degree (we used $\gamma = 0.3$ in Eq. 2.1) which is desirable in cases when output inhibition incorrectly suppresses valid inputs, hence hindering the correct output of the network. In the terms of an architecture, A-SRN$^+$ falls between A-SRN and SRN.

The last model, A-SRN$_{bck}$, which is shown in Fig. 2.3, uses an alternative, internal attentional mechanism, that is driven by direct connections from the hidden layer. It modulates the input similarly to A-SRN but allows a different flow of error during training by using an extra set of weights to separate the output representation (the scene interpretation) from the attentional information.

6.3.2 Network training

We have systematically looked for optimal network parameters which were then used in testing the models and performing comparisons as described below. The hidden layer of all networks was chosen to have 150 units, except the ESN where we used 300 reservoir neurons. Reservoir of ESN was initialized to have largest singular value $\sigma_{max} = 0.95$ and sparsity 10%. As an activation function for both hidden and output neurons we have used sigmoidal function, except ESN model where we used bipolar sigmoidal function for the reservoir neurons. Networks were trained with BPTT algorithm by propagating the error after each word (algorithm can be found in Chapter 2.2) , using the learning rate 0.01.

We have generated 10,000 scenes, each of which was associated with two events. Model's attention was driven by linguistic input to the single, major event of each situation. All generated events were consistent with the world, obeying semantic constraints. With each scene representation, a number of distractors (ranging from 0 to 3) was added to the input, taken from the pool of remaining agents/patients. Randomly chosen 70% of situations were used for training and the remaining 30% for testing. Data sets were distinguished by major events used in the scenes.

6.4 Model evaluation

In further text we will evaluate performance of various models trained for utterance-driven visual attention task and we present measures used in this evaluation. To investigate model's performance, we need to evaluate both components of network's output (OBJ and EV). For testing the accuracy of event level output $\mathbf{e}_{\text{out}}$ we decode corresponding output part (using the hidden-output weights of AAN) and count the percentage of correct decodings in the test set. Regarding of object-level output $\mathbf{c}_{\text{out}}$, we compare the output with all possible combinations of OBJ representations (in SOM), i.e. $\mathbf{c}_{\text{tgt}}$. Analogically, we count the percentage of matches (for both agents and patients). The standard cosine measure is applied for both EV and OBJ outputs. All measures can be evaluated after each word presented, to capture the progress during sentence processing. We looked at the output accuracy at the end of sentences and also on network's anticipatory behaviour, that is, its prediction of upcoming constituents during sentence processing (i.e. predicting an action when reading a subject word and predicting a patient when reading a subject or action words).

The illustration of a trained A-SRN during processing at the sentence *boy chases dog* is shown in Fig. 6.3, and corresponds to the scene in Fig. 6.1. OBJ-related graphs contain 8×8 units, EV-related graphs contain 48-dimensional vectors, reshaped to 8×6 matrix for convenience. On the right, OBJ input is the composition of various objects (including distractors), EV input is the superposition of two events. Both inputs are presented to the

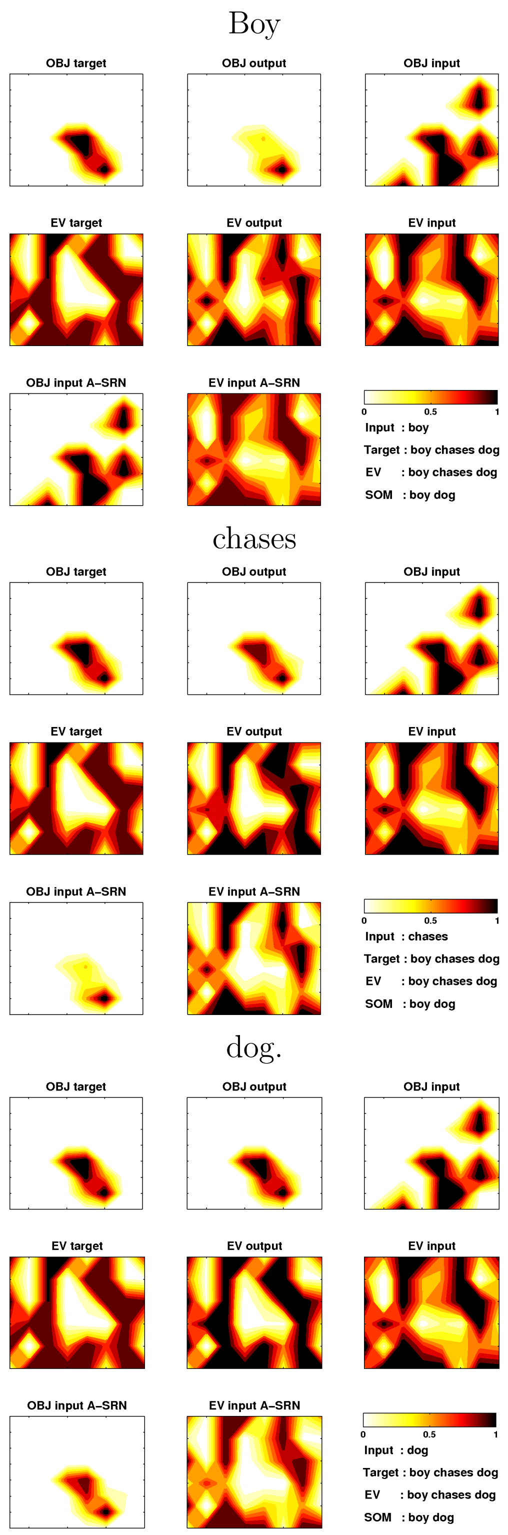


Figure 6.3: Example of A-SRN activation during sentence processing for sentence *Boy chases dog*. We can notice improvement of output activation compared to targets for both EV and OBJ output parts.

network at the sentence beginning. On the left, both targets comprise only information about the target event (and the pertaining objects). At the bottom, both input become overriden by the the feedback attentional mechanism that filters out irrelevant objects and non-target event information. Visual inspection of the network outputs (in the middle) reveals that they match well with both corresponding targets.

6.5 Performance measures

We explain all measures used in Tables 6.6–6.13. Symbol 'x' refers to the stage processing in a sentence (if x=1, the first word is as the input). All measures share the property that the closer the value to 1 (from below), the more accurate the value.

Acronym	Description
cos	cosine between the target situation vector and the network output (both OBJ and EV parts concatenated)
EV	quantifies the accuracy of network output by decoding it at sentence end; successful if both objects and action match the targets
EVa1	prediction of action when reading a subject; important measure since action cannot be retrieved from OBJ (unlike objects)
EVpx	prediction of the patient before the sentence end
EVa1_{W}	predictions of the possible actions from output after the first word; action is correctly decoded when it is consistent with the world (i.e. it exists in the corpus in the given context)
EVpx_{W}	predictions of possible patients; successful if consistent with the world (i.e. it exists in the corpus in the given context)
EVa1_{S}	action predictions; considered correct if the action was present in the current situation (in visual input)
EVpx_{S}	predictions of patients; assumed correctly decoded if it was present in current situation (in visual input)
OBJx	prediction of agent/patient pairs; successful if both objects match the target

Table 6.5: Quantitative measures used for evaluating model performance.

Measures starting with EV- are related to event-level while measures starting with OBJ-are related to object level. Event-level measures are calculated from $\mathbf{e}_{\text{out}}$, while object level measures are calculated from $\mathbf{c}_{\text{out}}$. The symbol $x \in \{1, 2\}$. For model evaluations, we have used the measures listed in Table 6.5.

6.6 Evaluation properties

Results in all tables refer to the testing data. We focused at three factors when evaluating model performance. First, we compared the accuracy of four models at the end of sentence; second, we manipulated the availability of the scene information during training and investigated its effect on model behavior; third, we looked at predictive properties of the model, i.e. the anticipation of upcoming consituents before the sentence end.

All results shown in tables are averages of 5 simulations, all with standard deviations below 0.02. Standard deviation was larger only for results with fully omitted visual scene inputs, with highest values reaching 0.1.

6.7 End-of-sentence behaviour

Model's ability to yield correct interpretation of the event, mediated by linguistic utterance, can be evaluated only at the end of sentence. Table 6.6 shows that all models have learned to generate correct output with high accuracy for both parts of its representation. SRN was observed to perform sufficiently what suggests that the feedback mechanism used in A-SRN-based models is not crucial for this relatively simple task. However, the feedback mechanism used in A-SRN$_{\text{bck}}$ improved the performance to nearly 100% accuracy.

Model	cos	EV	OBJ
SRN	0.995	0.985	0.986
A-SRN	0.981	0.899	0.949
A-SRN^{+}	0.986	0.949	0.976
A-SRN$_{\text{bck}}$	0.995	0.996	0.992
ESN	0.976	0.915	0.971

Table 6.6: Model performance with respect to the target event, evaluated at the end of sentence.

In the case of trained A-SRN, we have examined its behavior and found out that it might be the suboptimality of its strict attention mechanism that sometimes inhibits (via sigma-pi connection) the target objects/actions at the input, hence reducing the output accuracy towards the end of sentence. To test this hypothesis, we introduced the model A-SRN^{+} and its performance was observed to be expectedly better than A-SRN.

As we could see in chapter 5.5, it is crucial for echo-state networks to correctly initialize input representations. The correct scale of the representation can be applied to input weights $\mathbf{W}_{\text{in}}$ with similar effect. We have applied the scale to the input weights $\mathbf{W}_{\text{inL}}$ and $\mathbf{W}_{\text{inS}}$ and found out that the need of its correct initialization in this task is even more important, because of extensive input preprocessing and different scale of both types of inputs. Best results were achieved with ESN with 0% sparsity for both input matrices and scaling factor of visual input 0.1.

6.8 Restriction of situational input

We have restricted the availability of the visual input during training, either completely, or by randomly choosing 50% of sentences (in each training epoch). The purpose of this manipulation was twofold: to simulate the lack of visual input (for example, to simulate mere listening about the given event) but also to force the network to rely more on the linguistic pathway in predicting the output. The models were then tested on two types of novel inputs – those with and those without available visual inputs. Corresponding results are shown in Table 6.7.

The simulations reveal, that partial omission of scene inputs during training positively affects model accuracy, especially that of A-SRN. Interestingly, A-SRN yields a better performance also on testing data patterns with corresponding scene inputs, compared to the training mode with 100% availability of scene information (Table 6.6). The ESN achieved imperfect performance because of alternating visual inputs. Input matrices which are not trained, cannot provide sufficient dynamics of reservoir, necessary to represent arbitrary presence of visual input. To achieve best performance, we have rescaled visual input of ESN with factor 0.1.

The complete removal of the scene input during training lead to excessive bonding between visual contexts and spoken language resulting in good performance for data without visual scene input (see EV^{e} and OBJ^{e} in Table 6.7). However, when testing the network with available visual inputs, the results have deteriorated (both measures EV & OBJ), showing that the network does not have the ability to correctly comprehend the described event within the visual world. Because of the top-down attentional mechanism, A-SRN-based models could handle this type of testing much better, possibly taking advantage of the initial output representation evoked by the (sole) linguistic input and fed back as the scene input that eventually contributed to higher accuracy at the sentence end. On the other hand, the ESN was not able to cope with completely novel visual inputs and failed to comprehend described event. The output layer could not match novel states in reservoir[3] to the trained patterns, what caused low performance of the ESN. Best results for ESN were

[3]Affected by novel inputs, which were not modified by static input layer.

achieved with 90% sparsity for linguistic weight matrix and the scaling factor of linguistic input with value 10.

Model	%	cos	EV	OBJ	EV^e	OBJ^e
SRN	50	0.995	0.995	0.989	0.995	0.992
A-SRN	50	0.991	0.989	0.988	0.991	0.990
$A\text{-}SRN^+$	50	0.993	0.992	0.990	0.995	0.994
$A\text{-}SRN_{bck}$	50	0.997	0.998	0.994	1.000	1.000
ESN	50	0.980	0.920	0.980	0.790	0.900
SRN	0	0.929	0.504	0.627	0.999	0.997
A-SRN	0	0.963	0.769	0.823	0.998	0.994
$A\text{-}SRN^+$	0	0.947	0.671	0.688	1.000	0.994
$A\text{-}SRN_{bck}$	0	0.970	0.863	0.822	0.999	0.999
ESN	0	0.510	0.090	0.100	0.950	0.970

Table 6.7: Model performance with respect to the target event for 50% and 100% empty situation input, evaluated at the end of sentence. Performance was computed for the test data with full (EV, OBJ) and empty (EV^e, OBJ^e) scene input.

6.9 Anticipation of upcoming constituents

Tables 6.8–6.12 refer to the prediction accuracy (constituent anticipation) during sentence processing. All five models predict the target action (EVa1) with ∼50% accuracy (Table 6.8). These predictions are mostly consistent w.r.t. depicted scene (∼75%, Table 6.10) and almost always consistent with the world knowledge (∼97%, Table 6.9) .

Prediction of the patient can be assessed at two steps. Upon reading a subject (EVp1), the predictability of the patient is around 50% w.r.t. the target but grows over 80% w.r.t. for both world knowledge and the depicted scene. Prediction of a patient one step later (EVp2) grows to about 65% w.r.t. target (except ESN), to about 95% w.r.t. the world knowledge and to 85% w.r.t. the depicted scene. ESN model exhibited deteriorated performance, namely for the prediction of patient w.r.t target and depicted scene during presentation of the action.

Prediction at the level of agents and patients (OBJx) is slightly less accurate. Upon processing the first word, the accuracy of predicting both objects remains at around 50% (having agent accurate but the patient inaccurate), and only grows to ∼60% when processing the verb. Also for this task, the ESN provided worse prediction abilities as the other models.

Models with omitted scene-related inputs (Table 6.11) exhibit decreased prediction

Model	EVa1	EVp1	EVp2	OBJ1	OBJ2
SRN	0.522	0.575	0.706	0.501	0.625
A-SRN	0.503	0.510	0.645	0.452	0.588
A-SRN$^+$	0.491	0.517	0.667	0.484	0.608
A-SRN$_{\mathrm{bck}}$	0.498	0.545	0.697	0.479	0.597
ESN	0.489	0.447	0.553	0.252	0.302

Table 6.8: Network anticipation of upcoming constituents with respect to target.

Model	EVa1$_{\mathrm{W}}$	EVp1$_{\mathrm{W}}$	EVp2$_{\mathrm{W}}$
SRN	0.975	0.872	0.964
A-SRN	0.971	0.836	0.939
A-SRN$^+$	0.969	0.843	0.952
A-SRN$_{\mathrm{bck}}$	0.971	0.857	0.963
ESN	0.964	0.783	0.915

Table 6.9: Network anticipation accuracy with respect to world knowledge.

Model	EVa1$_{\mathrm{S}}$	EVp1$_{\mathrm{S}}$	EVp2$_{\mathrm{S}}$
SRN	0.765	0.841	0.883
A-SRN	0.742	0.806	0.856
A-SRN$^+$	0.732	0.815	0.864
A-SRN$_{bck}$	0.754	0.830	0.894
ESN	0.753	0.814	0.778

Table 6.10: Network anticipation accuracy with respect to consistency with the depicted scene.

ability because of missing visual scene information. When no situation inputs are presented during training, model does not rely on this type of information, thus ignoring it also for test set with visual information available. In addition, the prediction in the data set without the visual input was not achieved by any model.

To describe anticipation abilities of the models with restricted scene input in more detail, we have additionally examined their performance w.r.t. world knowledge. This method is similar with evaluation of prediction accuracy used in Chapter 5. According to Table 6.12, the usage of partly omitted scene inputs does not interfere with anticipation w.r.t. world knowledge, in opposite to anticipation w.r.t. target. In almost all cases the networks successfully predict suitable actions and patients during sentence processing. The smallest prediction ability was observed during prediction of more distants constituents (i.e. anticipation of patient during processing of the agent; measure EVp1$_W$).

Model	%	EVa1	EVp1	EVp2	OBJ1	OBJ2
SRN	50	0.509	0.521	0.681	0.449	0.549
A-SRN	50	0.475	0.456	0.639	0.423	0.506
A-SRN$^+$	50	0.492	0.477	0.664	0.454	0.569
A-SRN$_{\text{bck}}$	50	0.489	0.556	0.675	0.492	0.561
ESN	50	0.460	0.410	0.490	0.190	0.210
SRN	0	0.201	0.039	0.074	0.015	0.031
A-SRN	0	0.203	0.033	0.085	0.006	0.022
A-SRN$^+$	0	0.183	0.039	0.090	0.005	0.016
A-SRN$_{\text{bck}}$	0	0.201	0.042	0.080	0.015	0.020
ESN	0	0.013	0.050	0.060	0.010	0.010

Table 6.11: Network anticipation of upcoming constituents with respect to the target for models with omitted scene inputs (tested on data with available visual scene).

Model	%	EVa1$_W$	EVp1$_W$	EVp2$_W$	EVa1^e_W	EVp1^e_W	EVp2^e_W
SRN	50	0.975	0.851	0.965	0.999	0.608	0.954
A-SRN	50	0.971	0.819	0.951	0.999	0.592	0.938
A-SRN$^+$	50	0.970	0.838	0.958	1.000	0.601	0.854
A-SRN$_{\text{bck}}$	50	0.970	0.869	0.961	1.000	0.609	0.884
ESN	50	0.960	0.750	0.890	0.990	0.340	0.810
SRN	0	0.966	0.635	0.840	0.913	0.743	0.845
A-SRN	0	0.979	0.601	0.873	0.930	0.226	0.880
A-SRN$^+$	0	0.950	0.604	0.889	0.894	0.602	0.848
A-SRN$_{\text{bck}}$	0	0.964	0.671	0.862	0.917	0.208	0.841
ESN	0	0.800	0.690	0.720	0.990	0.630	0.890*

Table 6.12: Network anticipation of upcoming constituents with respect to world knowledge (tested on data with omitted scene inputs).

6.10 Processing situations with multiple events

To simulate a more realistic world environment, we created a data set with multiple events per visual scene. Each scene contained two or three events (with 50:50 ratio) and a random number of distractors. Similarly as in the previous text, the network task was to select the target event mediated by the utterance.

Tables 6.13 and 6.14 show that all models were able to process situations with multiple events, having surprisingly better performance at the end of sentences. Complexity of visual scene has caused that models had to rely on linguistic inputs resulting in similar behaviour as we could see in case of restricted scene input (Section 6.8). On the other

Model	cos	EV	OBJ
SRN	0.995	0.989	0.989
A-SRN	0.986	0.957	0.970
A-SRN$^+$	0.989	0.981	0.984
A-SRN$_{\text{bck}}$	0.995	0.997	0.993
ESN	0.975	0.918	0.962

Table 6.13: Model performance for multiple events per visual scene evaluated at the end of sentence.

hand, prediction accuracy suffers from multiple object and event possibilities, resulting in deteriorated performance. Anticipation w.r.t. world knowledge expectedly is not affected by the presence of multiple events but the anticipation w.r.t. depicted scene is aggravated, because of numerous object combination within the visual scene.

Model	EVa1	EVp1	EVp2	OBJ1	OBJ2
SRN	0.463	0.447	0.586	0.343	0.486
A-SRN	0.426	0.347	0.508	0.207	0.411
A-SRN$^+$	0.450	0.363	0.523	0.293	0.437
A-SRN$_{\text{bck}}$	0.447	0.430	0.558	0.348	0.460
ESN	0.438	0.380	0.469	0.163	0.199

Table 6.14: Network anticipation of upcoming constituents w.r.t. target for multiple events per visual scene.

Model	EVa1$_W$	EVp1$_W$	EVp2$_W$	EVa1$_S$	EVp1$_S$	EVp2$_S$
SRN	0.980	0.861	0.954	0.776	0.781	0.846
A-SRN	0.975	0.798	0.938	0.708	0.698	0.795
A-SRN$^+$	0.977	0.806	0.942	0.766	0.719	0.811
A-SRN$_{\text{bck}}$	0.978	0.843	0.953	0.769	0.789	0.842
ESN	0.973	0.770	0.913	0.769	0.814	0.776

Table 6.15: Network anticipation of upcoming constituents w.r.t. world knowledge and depicted scene for multiple events per visual scene.

6.11 Hidden-layer activations

If the network is able to correctly predict output, this ability should imply some organization of the network's internal representations at the hidden-layer. We performed an analysis of the hidden representations, using the traditional technique, first presented in Elman (1990). That is, the training data was again presented to the trained network in a single sweep, the hidden activation were recorded, and then reordered with respect to the same input word, and averaged over contexts. Like Elman, we could observe some degree of internal organization between words, albeit to a lesser degree. However, there two differences. First, out network was not trained on a next-word prediction task but shooting at a static target. Second, out linguistic input is modulated (and noised) by situational inputs. The example of a hierarchical cluster diagram is shown in Fig. 6.4. The order in other three models was somewhat less evident.

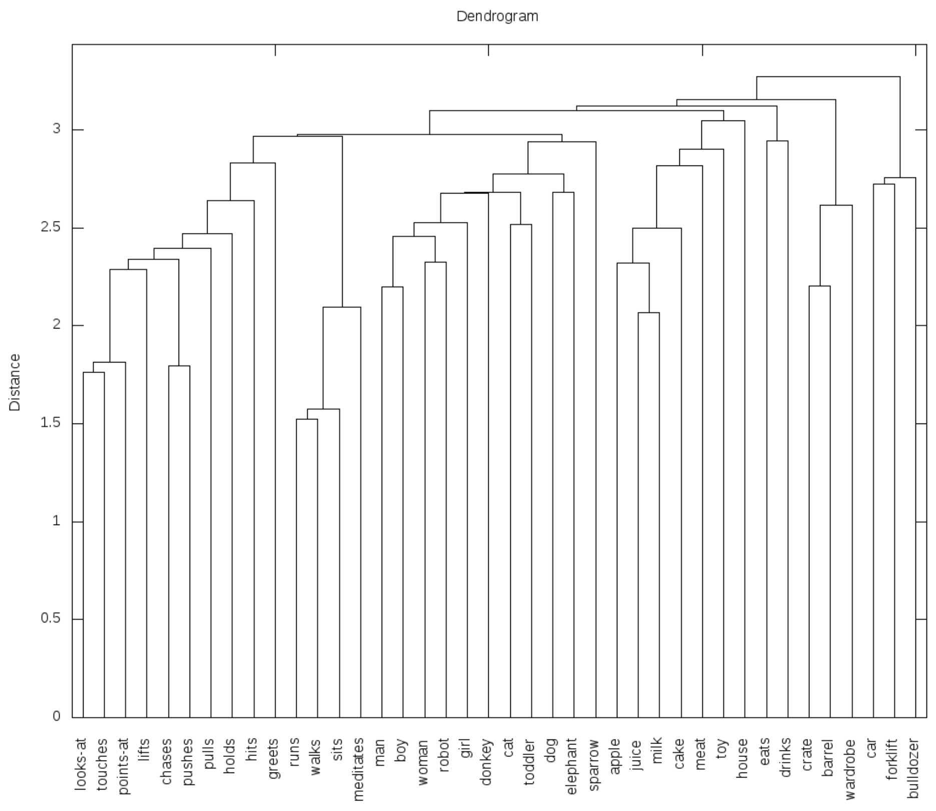


Figure 6.4: Hierarchical cluster diagram of hidden-unit activation vectors of a trained A-SRN$_{bck}$ model with the completely available scene input.

6.12 Analysis of attentional mechanism

To analyse attentional mechanism more deeply, we compared altered sigma-pi scene input with the current target. Sigma-pi connection, driven by network output, filters irrelevant objects within visual scene, what causes larger correlation between altered input and target event vectors. To test this hypothesis, we have used the standard cosine measure, comparing event part (EV), object part (OBJ) of the mentioned vectors and their concatenation (EV-OBJ). As we can see in Figures 6.5-6.7, the A-SRN and A-SRN$^+$ models indeed increase similarity between mentioned vectors. In case of A-SRN, the average cosine decrease during processing of the second word, what is consistent with our observation that output attentional mechanism can misguide network by activating incorrect event. This issue is resolved in A-SRN$^+$ model. The results in Figure 6.7 shows that the A-SRN$_{\text{bck}}$ uses

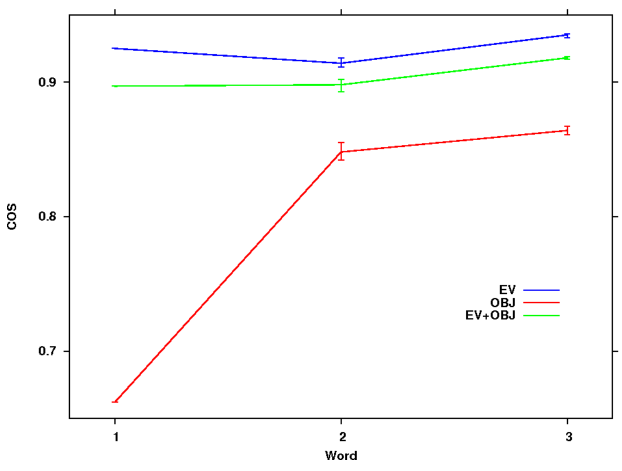


Figure 6.5: Cosine between modified sigma-pi scene input vector and target visual scene vector during sentence processing with A-SRN network. During processing of the first word, the network uses unmodified input.

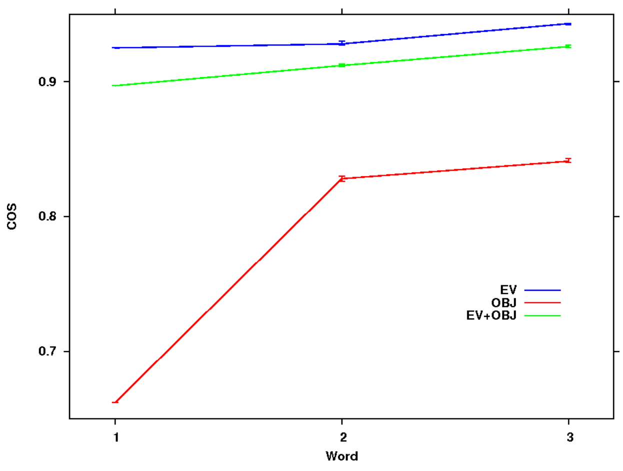


Figure 6.6: Cosine between modified sigma-pi scene input vector and target visual scene vector during sentence processing with A-SRN$^+$ network.

different type of attentional mechanism, which does not predictively filter the objects and events in input visual scene. In fact, it inhibits the visual input significantly (Figure 6.8), resulting in decreased cosine between input and target vectors (Figure 6.7). The network probably uses its hidden layer more extensively to memorize the visual scene and in later sentence positions focus on the language-output mapping since network outputs are more accurate (as shown in previous sections).

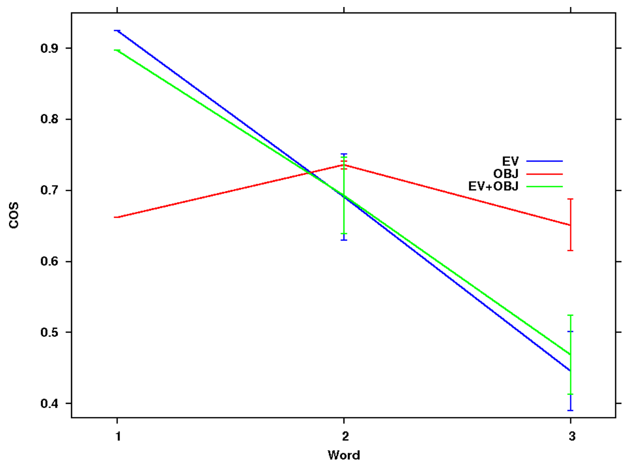


Figure 6.7: Cosine between modified sigma-pi scene input vector and target visual scene vector during sentence processing with A-SRN$_{bck}$ network.

6.13 Discussion

This chapter describes situated language processing using the visual world paradigm. We compared SRN and ESN with several recurrent neural network models with an explicit attentional mechanism to appreciate the role of the feedback in sentence comprehension task. All models can almost perfectly learn to generate the end-of-sentence representation that is interpreted as the sentence meaning in the visual context. Having read the sentence, each network correctly selects the target scene event and its corresponding constituents (agent/patient). To a certain degree, all networks also demonstrate predictive behavior reflected by the ability to anticipate upcoming constituents. SRN performs expectedly very well, but we have shown that adding an explicit attentional mechanism (in A-SRN$_{\mathrm{bck}}$) results in slight improvement of the performance. The availability of the attentional mechanism helps A-SRN models to perform better on testing data with and without the scene information when trained on inputs with restricted scene information. From the cognitive perspective, A-SRN's attentional mechanism helps the network focus on the relevant scene event and incorporates into the model the visual attention system on an abstract level, and reveals similar anticipatory shifts in visual attention that have

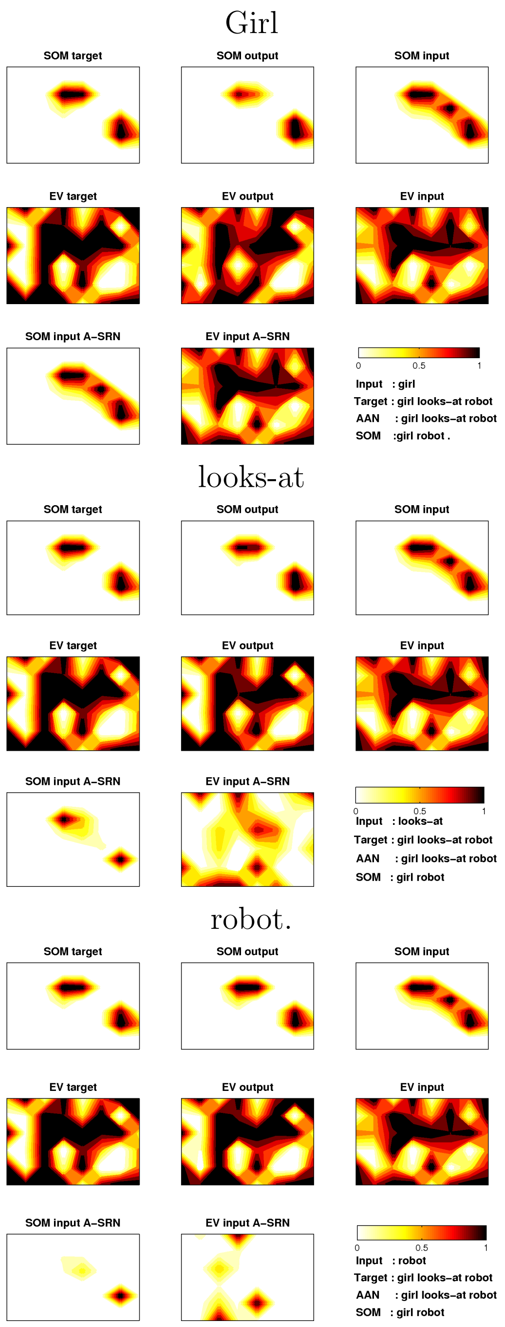


Figure 6.8: Example of A-SRN$_{\text{bck}}$ activation during sentence processing for sentence *Girl looks-at robot*. We can notice significant inhibition of the scene input with the activation of the explicit hidden attentional layer.

been found using the visual world paradigm (Knoeferle et al., 2005; Knoeferle and Crocker, 2006). One exception is the A-SRN$_{\mathrm{bck}}$ model which probably uses its hidden layer more extensively to memorize the visual scene. Its attentional mechanism inhibits the visual input in later sentence positions, probably using the language-output mapping to process the last words of sentence more effectively. On the other side, the ESN have exhibited deteriorated performance in many subtasks which was caused in most cases by inability to cope with alternating inputs.

We have shown that models are able to process linguistic utterances also without visual data but adding scene input helps network to correctly emphasize described event within visual world and enables anticipation of particular upcoming event constituents. A-SRN-based models differ crucially from CiaNet (Mayberry et al., 2009) that served as our motivation, in their potential to deal with complex visual scenes containing more than two events. This property allows describing more realistic world scenes and deal with complex (possibly recursive) sentences with multiple relations between their constituents. Regarding the world complexity, we expect that the benefits of the A-SRN model (i.e. anticipation of objects in the scene) may in fact increase as the knowledge of the network scales up, that is, when there's a larger difference between what the network learns during training, and what is currently depicted when processing a given sentence.

Chapter 7

Conclusion

This thesis covers several subtasks of natural language processing. In the first part (Chapter 4) we have explored the non-adjacent dependencies in the artificial syllables language, showing that also echo-state networks are able to satisfy variability hypothesis. Moreover, our modelling confirmed that ESN is able to find the places of high entropy within continuous sequence of syllables, therefore splitting the sequence into words.

In the second part (Chapter 5) we have focused on higher language processing. We have trained echo-state networks on next word prediction task within recurrent sentences of simplified English language. We have extended the grammar acquisition tasks performed in Tong et al. (2007) for more complex input language. Although, input language used in Tong et al. (2007) has rich recurrent structure, lexicon it use has only 24 input words. We have extended this language to use lexicon with 96 words and showed that ESN is able to process also more complex languages. However, its performance has deteriorated. Our assumption is that the combination of recurrent structure with larger lexicon size creates too large corpus, what prevents ESN to successfully process all possible sentence combinations.

To increase performance of echo state-networks we have used various input representations in addition to frequently used localist encoding. These comprise random distributed representations and representations obtained by corpus preprocessing. ESN$^+$ model, introduced by Bullinaria and Levy (2007), use simple analysis of word neighbourhood, while WCD representations (Li et al., 2004) take into consideration also larger contexts and order of the neighbouring words. Our results confirm findings from Frank and Čerňanský (2008), that using distributed representations obtained by preprocessing of corpus increases performance of ESN. However, deeper analysis conducted in Švantner and Farkaš (2009a), shows that ESN$^+$ model takes advantage also from larger scale of generated representations. Normalized ESN$^+$ and WCD representations exhibit comparable performance to the one observed in random distributed and localist representation. The increase of scale of representations distinguish the performance of ESN$^+$ and WCD compared to random distributed and localist models. These findings led us to conclusion that performance of echo-state networks depends also on scale of presented inputs and for different tasks we need to scale input independently. This fact is logical, since input weights of ESN are not trained and network cannot adjust them to its needs. On the other hand, the effect of larger neigbourhood contexts and the order of words, have shown surprisingly only subtle impact on ESN performance, even for the larger input scaling. In the further work, we have generated reasonable input representations using the multiple executions of the ESN. The echo state network which used these representations achieved similar performance to ESN$^+$ and ESN$_{WCD}$ models.

In the third part (Chapter 6) we have explored the impact of visual scene on language processing and modelled attention incurred within visual world. Recurrent neural networks (namely SRN, A-SRN and ESN) have been trained to filter described part of visual scene and provide its representation on output. All networks showed almost perfect end-of-sentence performance, correctly choosing described event and its objects. Our implementation allows processing of more complex visual scene (as opposed to CiaNet; Mayberry et al., 2009), which comprises multiple concurrent events. None of the models had any problems with this modification either. To model off-line description of visual event, we have removed the visual input in randomly selected scenes, what resulted in better end-of sentence behaviour. This can be explained by reinforced utterance-output mapping during removal of visual scenes. However, removal of all visual inputs caused that networks have exhibited deteriorated performance during presence of the scene in testing data. In addition, visual input allows to predict particular action and objects of described event even before end of utterance. It eases language processing and shows that models are in addition to network shown in Chapter 5 able to filter object within complex visual scene, focusing attention on important part of the scene.

Echo-state network did not successfully model all of previously mentioned behaviour. Namely in prediction task and partly present visual input environment was its performance deteriorated. ESN was not able to cope with variable visual input, even after proper input scaling. To explicitly model attentional mechanism we have introduced A-SRN-based models. These have shown comparable performance to one of simple recurrent network in all previously mentioned tasks, exhibiting superior performance during omitted visual input. In addition, A-SRN-based models offer better characterization of attentional mechanism, exhibiting all of its desired properties.

Resumé

Táto dizertačná práca sa zaoberá spracovaním prirodzeného jazyka pomocou rekurentných neurónových sietí. V prvej časti upriamujeme pozornosť na úlohu predikcie nasledujúcich symbolov v prirodzenom texte. V krátkosti rozoberáme aktuálny stav tematiky a opisujeme architektúry neurónových sietí, ktoré sa pri spracovávaní prirodzeného jazyka používajú. Medzi najpoužívanejšie patria jednoduchá rekurentná sieť (SRN, obr. 2.1) a sieť s echo stavmi (ESN, obr. 2.4).

Kapitola 4 je venovaná predikčnej úlohe na úrovni slabík umelého jazyka. Na jednoduchom jazyku, pozostávajúcom zo zreťazených trojslabičných slov, sme modelovali predikciu nasledujúcej slabiky, ktorá využíva susedné aj nesusedné závislosti medzi jednotlivými slabikami (Farkaš a Švantner, 2007). Ukázali sme, že sieť s echo stavmi je schopná správne rozdeliť slová na miestach s vysokou entropiou. Sieť bola navyše schopná zohľadniť aj nesusedné závislosti medzi slabikami pri veľkých výplniach, umožňujúc generalizáciu (použitie známych závislostí aj pri nových výplniach).[1] Tým sme čiastočne potvrdili hypotézu variability, ktorá hovorí, že ľudia dokážu generalizovať v prípade veľmi vysokej alebo nulovej variability slabík (výplne) medzi nesusednými slabikami.

V kapitole 5 sme modelovali spracovávanie viet a akvizíciu gramatiky pomocou sietí s echo stavmi (Švantner a Farkaš, 2009b). Model bol trénovaný na úlohe predikcie ďalšieho slova vo vetách dvoch jednoduchých prirodzených jazykov, vychádzajúcich z angličtiny. Ako základnú dátovú množinu sme mierne upravili jazyk (Tong a spol., 2007), ktorý má síce bohatú rekurzívnu štruktúru, ale využíva iba málo slov. Z tohto dôvodu sme ho rozšírili, pridaním väčšieho slovníka a ukázali sme, že sieť s echo stavmi je schopná spracovávať aj o niečo zložitejšie jazyky (čo do veľkosti slovníka). Musíme však podotknúť, že jej úspešnosť sa pri použití zložitejšieho jazyka znížila. Predpokladáme, že zväčšenie slovníka v kombinácii so zložitou rekurentnou štruktúrou viet, viedlo k vytvoreniu množstva vetných kombinácií a sieť nebola schopná správne zovšeobecňovať z dôvodu, že sa počas trénovania stretla iba s malou časťou z nich.

Pre zvýšenie úspešnosti siete s echo stavmi sme pre jej vstupy vyskúšali rôzne reprezentácie. Okrem lokalistických a náhodných distribuovaných kódov sme použili

[1]Napríklad, ako výplň medzi závislými slabikami '*ľudia*' a '*jú*' vo vete '*<u>Ľudia</u> sa pozera<u>jú</u> na oblohu.*' považujeme časť vety '*sa pozera*'.

reprezentácie, ktoré vznikli štatistickým predspracovaním textu. Model ESN$^+$, ktorý bol navrhnutý v Frank a Čerňanský (2008), používa jednoduchú analýzu, založenú na početnosti spoločných výskytov slov v rámci textu. WCD reprezentácie (Li a spol., 2004) sú získané veľmi podobnou metódou, avšak zohľadňujú aj poradie v rámci spoločných výskytov a umožňujú použitie väčších kontextov medzi skúmanými slovami. Naše výsledky potvrdzujú zistenia z Frank a Čerňanský (2008), že použitie distribuovanej reprezentácie získanej štatistickým predspracovaním textu (ESN$^+$ a WCD reprezentácie) zvyšuje úspešnosť sietí s echo stavmi. Hlbšia analýza však ukázala (Švantner a Farkaš, 2009a), že zvýšenie úspešnosti sietí s echo stavmi bolo z časti spôsobené zmenou škály vstupov. Po normalizácii vstupov (škála vstupov bola znížená) bola úspešnosť sietí s echo stavmi používajúcich reprezentácie získané pomocou štatistického predspracovania textu porovnateľná s modelom využívajúcim lokalistické a náhodné distribuované kódy. Tieto zistenia nás viedli k záveru, že úspešnosť sietí s echo stavmi záleží okrem iného aj na škále vstupných reprezentácií a pre rôzne úlohy potrebujeme zistiť aj správnu škálu vstupov. Tento fakt je logický, keďže sa vstupné váhy v sieťach s echo stavmi netrénujú, a teda ich sieť nie je schopná upraviť pre svoje potreby. Ostatné variácie vstupných reprezentácií nepriniesli požadované zvýšenie úspešnosti sietí s echo stavmi. Medzi skúmanými bola veľkosť kontextu medzi závislými slovami a poradie slov. V ďalšom výskume (Švantner, 2010) sme vytvorili sémantické reprezentácie vstupných slov pomocou viacnásobného spracovania textu pomocou siete s echo stavmi. Ukázali sme, že tieto reprezentácie umožňujú dosahovať podobné výsledky ako reprezentácie vytvorené štatistickým predspracovaním textu. Ich výhodou je, že môžu byť modelované iba pomocou neurónových sietí, čo je biologicky prijateľnejšie.

V kapitole 6 sme neskúmali jazyk iba ako samostatnú entitu ale študovali sme ho v kontexte vizuálnej percepcie. Opísali sme vplyv vizuálnej scény na spracovávanie prirodzeného jazyka a modelovali sme upriamovanie pozornosti na objekty vizuálneho sveta. Úlohou rekurentných neurónových sietí bolo spoznať na opisovanú udalosť v rámci vizuálnej scény a poskytnúť jej reprezentáciu na výstupe (Švantner a spol., 2011b). Jednotlivé modely sme trénovali na vstupných dátach s vizuálnymi scénami, ktoré obsahovali dve alebo tri súčasné udalosti. Udalosti boli zakódované pomocou autoasociačnej siete a ich kódy boli sčítané po zložkách, aby vytvorili udalostnú časť reprezentácie scény. Všetky objekty scény[2] boli upravené pomocou samoorganizujúcej sa siete, ktorá mala za úlohu vytvoriť podobné reprezentácie pre objekty s rovnakými sémantickými vlastnosťami (obr. 6.2). Výsledné reprezentácie boli následne sčítané po zložkách a vytvorili objektovú časť reprezentácie scény. Jazykový vstup opisujúci cieľovú udalosť bol predstavený postupne v podobe lokalisticky reprezentovaných slov. Všetky siete boli na konci opisu schopné vybrať správnu udalosť a určiť všetky objekty, ktoré

[2]Počítajúc aj distraktory, ktoré slúžili na skomplikovanie scény.

sa v nej podieľali. Naša implementácia, na rozdiel od Mayberry a spol. (CiaNet; 2009), umožňuje modelovať pozornosť aj v zložitejších vizuálnych scénach, ktoré pozostávajú z viacerých súčasných udalostí.

V ďalšom výskume sme z časti vstupných dát odstránili vizuálne vstupy. Motiváciou tohto kroku bola snaha o modelovanie opisu aktuálne neprítomnej scény. Úspešnosť modelov sa zvýšila pri koncoch viet, čo bolo spôsobené posilnením závislostí medzi jazykovým opisom jednotlivých scén a ich vizuálnou reprezentáciou počas trénovania. Vynechanie vizuálnych vstupov zo všetkých scén však spôsobilo zníženie úspešnosti všetkých modelov, lebo dochádzalo k prílišnému naviazaniu výstupných reprezentácií na opis. Opätovné predstavenie vizuálnych vstupov v testovacej množine zvýšilo chybovosť modelov. Siete boli schopné do určitej miery predikovať budúce akcie a objekty v rámci aktuálne opisovaných udalostí. Táto schopnosť zefektívňuje spracovanie jazyka a ukazuje, že rekurentné neurónové siete sú schopné odfiltrovať nepodstatné objekty z vizuálnej scény, a preto sú vhodné na modelovanie pozornostného mechanizmu.

Siete s echo stavmi nedokázali úspešne modelovať všetky vyššie spomínané podúlohy (Švantner, 2011). Zníženú úspešnosť zaznamenali menovite pri čiastočne prítomnom vizuálnom vstupe a pri predikcii. Siete s echo stavmi sa totiž nie sú schopné vysporiadať so zmenou časti vstupov, dokonca ani po správnom preškálovaní ich vstupných reprezentácií.

Okrem štandardných modelov (SRN, ESN) sme pre túto úlohu vyvinuli sieť A-SRN (obr. 2.2), ktorá na modelovanie pozornostného mechanizmu používa prídavné rekurentné spojenie, priamo ovplyvňujúce vstupy siete (Švantner a spol., 2011b). A-SRN sieť dosahovala podobnú úspešnosť ako SRN vo všetkých spomínaných úlohách a ukázala takmer bezchybné výsledky pri podúlohe s čiastočne prítomnými vizuálnymi vstupmi. Jej rozšírenie, model A-SRN$_{bck}$ (Švantner a spol., 2011a), ktorého architektúru môžeme vidieť na obr. 2.3, používa jednu skrytú vrstvu navyše, ktorá predstavuje explicitný vnútorný pozornostný mechanizmus. Zložitejšia architektúra umožňuje modelu A-SRN$_{bck}$ dosahovať lepšie výsledky takmer vo všetkých podúlohách. Siete A-SRN detailnejšie charakterizujú pozornostný mechanizmus a modelujú všetky jeho požadované vlastnosti.

Bibliography

Bullinaria, J. & Levy, J. (2007). Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, **39**, 510–526.

Christiansen, M. & Chater, N. (1999a). Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, **23**(2), 157–205.

Christiansen, M. H. & Chater, N. (1999b). Connectionist natural language processing: the state of the art. *Cognitive Science*, **23**, 417–437.

Cleeremans, A., Destrebecqz, A., & Boyer, M. (1998). Implicit learning: news from the front. *Trends in Cognitive Sciences*, **2(10)**, 406–416.

Cottrell, G. W., Munro, P., & Zipser, D. (1989). Image compression by back propagation: an example of extensional programming. *Models of Cognition: A Review of Cognition Science.*

Černák, M. (2005). *Učenie nesusedných závislostí pomocou rekurentných neurónových sietí.* Master's thesis, Fakulta matematiky, fyziky a informatiky, Univerzity Komenského.

Čerňanský, M. & Makula, M. (2007). Spracovanie postupností symbolov pomocou esn sietí. In *Kognícia a umelý Život VII*, pages 93–98. Vydavateľstvo STU.

Čerňanský, M., Makula, M., & Ľ. Beňušková (2007). Organization of the state space of a simple recurrent network before and after training on recursive linguistic structures. *Neural Networks*, **20**(2), 236 – 244.

Elman, J. (1990). Finding structure in time. *Cognitive Science*, **14**, 179–211.

Elman, J. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, **7**, 195–225.

Elman, J. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, **48**(1), 71–79.

Farkaš, I. (2009). Learning nonadjacent dependencies with a recurrent neural network. In M. K. et. al (Eds.), editor, *International Conference on Neural Information Processing Systems ('2008)*, Lecture Notes in Computer Science, pages 292–299. Springer.

Farkaš, I. & Crocker, M. (2006). Recurrent networks and natural language: exploiting self-organization. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, pages 1275–1280, Hillsdale, NJ. Lawrence Erlbaum.

Farkaš, I. & Pokorný, M. (2009). Investigating systematicity in the linear RAAM neural network. In K. J. Mayor, N. Ruh, editor, *Connectionist Models of Behaviour and Cognition*, volume II, pages 217–228.

Farkaš, I. & Švantner, J. (2007). Učenie nesusedných závislostí pomocou rekurentných neurónových sietí. *Kognícia a Umelý Život*, **VII**, 389–394.

Frank, S. (2006a). Learn more by training less: systematicity in sentence processing by recurrent networks. *Connection science*, **18**(3), 287–302.

Frank, S. (2006b). Strong systematicity in sentence processing by an echo-state network. In *Proceedings of ICANN, Part I, Lecture Notes in Computer Science*, volume 4131, pages 505–514. Springer.

Frank, S. & Čerňanský, M. (2008). Generalization and systematicity in echo state networks. In *Cognitive Science*, volume 30, pages 733–738.

Gómez, R. (2002). Variability and detection of invariant structure. *Psychological Science*, **13**(5), 431–436.

Harnad, S. (1990). The symbol grounding problem. *Physica D*, pages 335–346.

Huettig, F., Rommers, J., & Meyer, A. S. (2011). Using the visual world paradigm to study language processing: A review and critical evaluation. *Acta Psychologica*.

Jaeger, H. (2001). The "echo state" approach to analyzing and training recurrent neural networks. Technical report gmd 148, German National Research Center for Information Technology.

James, D. & Miikkulainen, R. (1995). Sardnet: a self-organizing feature map for sequences. In *Advances in Neural Information Processing Systems 7*, pages 577–84. MIT Press.

Kamide, Y., Altmann, G., & Haywood, S. (2003). Prediction in incremental sentence processing: Evidence from anticipatory eye movements. *Journal of Memory and Language*, **49**, 133–156.

Karlsson, F. (2007). Constraints on multiple center embedding of clauses. *Journal of Lingustics*, **43**, 365–392.

Knoeferle, P. & Crocker, M. (2006). The coordinated interplay of scene, utterance, and world knowledge: Evidence from eye-tracking. *Cognitive Science*, **30**, 481–529.

Knoeferle, P., Crocker, M., Scheepers, C., & Pickering, M. (2005). The inuence of the immediate visual context on incremental thematic role-assignment: Evidence from eye-movements in depicted events. *Cognition*, **95**, 95–127.

Kohonen, T. (1990). The self-organizing map. In *Proceedings of the IEEE*, volume 78 (9), pages 1464–1480.

Lawrence, S., Lee Giles, C., & Fong, S. (2000). Natural language grammatical inference with recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering*, **12**(1), 126–140.

Li, P., Farkaš, I., & MacWhinney, B. (2004). Early lexical acquisition in a self-organizing neural network. *Neural networks*, **17**, 1345–1362.

Lukosevicius, M. & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, **3**(3), 127 – 149.

Mayberry, M. & Miikkulainen, R. (2003). Incremental nonmonotonic parsing through semantic self-organization. In *Proceedings of the 25th Annual Conf. of the Cognitive Science Society*, Mahwah, NJ. Erlbaum.

Mayberry, M., Crocker, M., & Knoeferle, P. (2009). Learning to attend: A connectionist model of situated language comprehension. *Cognitive Science*, **33**, 449–496.

Mintz, T., Newport, E., & Bever, T. (2002). The distributional structure of grammatical categories in speech to young children. *Cognitive Science*, **26**, 393–424.

Newport, E. (1990). Maturational constraints on language learning. *Cognitive Science*, **34**, 11–28.

Onnis, L., Christiansen, M., Chater, N., & Gómez, R. (2003). Reduction of uncertainty in human sequential learning: Evidence from artificial language learning. In *Proceedings of the 25th Annual Conference of the Cognitive Science Society*, pages 886–891, Mahwah, NJ. Lawrence Erlbaum.

Onnis, L., Monaghan, P., Christiansen, M., & Chater, N. (2004). Variability is the spice of learning, and a crucial ingredient for detecting and generalizing in nonadjacent dependencies. In *Proceedings of the 26th Annual Conference of the Cognitive Science Society*, pages 1047–1052, Mahwah, NJ. Lawrence Erlbaum.

Peña, M., Bonatti, L., Nespor, M., & Mehler, J. (2002). Signal-driven computations in speech processing. *Science*, **298**, 604–607.

Rohde, D. & Plaut, D. (2003). Connectionist models of language processing. *Cognitive Studies*, **10**(1), 10–28.

Roy, D. (2005). Grounding words in perception and action: computational insights. *Trends in Cognitive Sciences*, **9**, 389–396.

Rumelhart, D. & McClelland, J. (1986). *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, volume 2, chapter On learning the past tenses of English verbs, pages 216–271. MIT Press, Cambridge, CA.

Rumelhart, D., Hinton, G., & Williams, R. (1986). *Learning internal representations by error propagation*, volume 1, pages 318–362. MIT Press, Cambridge, MA.

Saffran, J. (2002). Constraints on statistical language learning. *Journal of Memory and Language*, **47**, 172–196.

Siskind, J. M. (2001). Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research*, **15**, 31–90.

Spivey, M., Tanenhaus, M., Eberhard, K., & Sedivy, J. (2002). Eye-movements and spoken language comprehension: Effects of visual context on syntactic ambiguity resolution. *Cognitive Psychology*, **45**, 447–481.

Švantner, J. (2010). Získavanie distribuovaných reprezentácií slov pomocou neurónových sietí s echo stavmi. *Kognícia a Umelý Život*, **X**, 375–379.

Švantner, J. (2011). Modelovanie vizuálnej pozornosti riadenej jazykom pomocou neurónových sietí s echo stavmi. *Kognícia a Umelý Život*, **XI**, 273–278.

Švantner, J. & Farkaš, I. (2009a). Investigating distributed representations in grammar acquisition with echo state networks. In *International Conference on Informatics*, volume 10, pages 265–270.

Švantner, J. & Farkaš, I. (2009b). Učenie gramatických závislostí pomocou neurónovej siete s echo stavmi. *Kognícia a Umelý Život*, **IX**, 319–325.

Švantner, J., Farkaš, I., & Crocker, M. (2011a). Modeling utterance-driven visual attention during situated comprehension. Submitted to *Neural Network World.*

Švantner, J., Farkaš, I., & Crocker, M. (2011b). Modeling utterance-mediated attention in situated language comprehension. In *Proceedings of the 33th Annual Conference of the Cognitive Science Society*, Mahwah, NJ. Erlbaum. Accepted.

Tanenhaus, M., Spivey-Knowlton, M., Eberhard, K., & Sedivy, J. (1995). Integration of visual and linguistic information in spoken language comprehension. *Science*, **268**, 1632–1634.

Tiňo, P., Farkaš, I., & van Mourik, J. (2006). Dynamics and topographic organization in recursive self-organizing map. *Neural Computation*, **18**, 2529–2567.

Tong, M. H., Bickett, A. D., Christiansen, E. M., & Cottrell, G. W. (2007). Learning grammatical structure with echo state networks. *Neural Networks*, **20**(3), 424–432.

Vančo, P. & Farkaš, I. (2010). Experimental comparison of recursive self-organizing maps for processing tree-structured data. *Neurocomputing*, **73**, 1362–1375.

Voegtlin, T. (2002). Recursive self-organizing maps. *Neural Networks*, **15**(8-9), 979–992.

Williams, R. & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, **1**, 270–280.

Yu, C., Ballard, D., & Aslin, R. (2005). The role of embodied intention in early lexical acquisition. *Cognitive Science*, **29**, 961–1005.