Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Word2Vec and Echo State Network For Thematic Role Assignment

Master Thesis
at the Research Group Knowledge Technology, WTM
Prof. Dr. Stefan Wermter

Department Informatik
MIN-Fakultät
Universität Hamburg

Submitted by
Surender Kumar
on
30.04.2013

Evaluators: Prof. Dr. Stefan Wermter
Dr. Sven Magg

Surender Kumar
Matriculation Number: 6519753
Kaemmererufer 13
22303 Hamburg

Abstract

Humans have a remarkable capability of acquiring language and in particular more than one languages. More interestingly they learn it within the same neural computing substrate. But how does the structure of a sentence is mapped to its meaning within the brain is still an open issue? To understand this process of language acquisition, several neural language models were proposed. Most of the existing model are either using syntactic parse trees to create handcrafted features or using localist vector representation of words to process the sentences. Hinaut et al. [55] proposed a $\theta RARes$ neural language acquisition model, for thematic role assignment which learned the thematic roles of the input sentences purely from the grammatical structure of the sentences. They also treated the words in the sentence as discrete atomic symbols which do not carry any semantic information about the words i.e. localist word representation was used. This thesis, proposes an end-to-end neural language model, in an extension to $\theta RARes$ model, which takes into account the semantics of the words and temporal aspect of the input sentences. The model contains a word2vec unit, which generates the distributed vector representation of words which captures the semantic and syntactic relationships along with a recurrent neural network namely, Echo state Network, with fixed random connections, which is capable of modeling the sequential data. Thus the model takes a raw sentence as an input which is processed word by word and as an output, the meaning of an input sentence is obtained. Our results shows that the proposed model performs better than the $\theta RARes$ model on thematic role assignment task.

Zusammenfassung

Hier die deutsche Zusammenfassung einfügen (notwendig).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Thematic Role Assignment (TRA) is a supervised learning problem which aims to identify events and its participants in a sentence and determine *"Who did what to whom"*. In other words assigning roles to words (arguments) in a sentence with respect to a verb (predicate). The role typically includes an agent, object, recipient, etc. For example, in the sentence *"the dog that gave the rat to the cat was hit by the man"*, the first noun *'dog'* is the agent of the verb *'gave'* and object of the verb *'hit'*. In Natural Language Processing terminology (NLP) the problem is studied under the name of Semantic Role Labelling (SRL). Hence, TRA or SRL is a form of simplistic semantic parsing which aims to determine the predicate-argument structure for a verb in the given sentence [55]. Understanding the semantics of the text plays an important intermediate step in a wide range of real-world applications such as machine translation [31], information extraction [5], sentiment analysis [54], document categorization [39], human-robot interaction [21, 5] etc.

1.1 Previous Work

Many successful traditional systems consider SRL as a multiclass classification problem and use linear classifier such as Support Vector Machines (SVM) to tackle the problem [30, 42, 41]. These systems are based on pre-defined feature templates derived from syntactic information obtained by parsing and producing parse trees of the sentences in the training corpus. However, in a study it was found that the use of syntactic parser certainly leads to degradation of thematic roles predictions [41]. Also, designing of feature templates needs a lot of heuristics and is a time-consuming process. The pre-defined features are often required to be iteratively modified depending on how the system performs with the selected features. The feature templates are often required to be re-designed when the task is to be carried out in different languages, corpora or when the data distribution is changed [55].

In order to evade manual feature engineering, SRL task was additionally endeavored with neural network models. Collobert et al. [14] first attempted to build an end-to-end system without using parsing and using a neural model with

word embeddings layer, Convolutional Neural Network (CNN) and Conditional Random Field (CRF). The model was less successful as CNN cannot employ long-term dependencies within a sentence since it can only take into account the words in limited context [55]. However, to increase the model performance they also resorted to using syntactic features by using parse trees of Charniak parser [11].

Recurrent Neural Networks (RNN) has also been used for a wide range of NLP tasks and also recently with Echo State Network (ESN), a variant of RNN. The tasks performed were diverse from predicting the next word given the previous words to learning grammatical structures [51]. A RNN makes use of sequential information and acts as a memory unit and captures the information processed in the past [17]. The ESN has several advantages over simple RNNs. First, ESNs are capable of modeling long-term dependencies in the sentence. Second, while processing long sequence, the gradient parameter vanishes or explodes in simple RNNs [7]. Third, unlike simple RNNs, ESNs are computationally cheap as the recurrent layer (reservoir) in ESN is randomly initialized and only the connections from recurrent layer to read-out layers are learned [28, 32]. These advantages of ESN over RNN makes it an excellent choice to be used for the TRA task.

Hinaut et al. [20] proposed a generic neural network architecture, $\theta RARes$ model, using Recurrent Neural Network based on reservoir computing approaches, namely Echo State Network to solve TRA task. The proposed architecture models the language acquisition in the brain and provides a robust and scalable implementation on robotics architecture [20, 21]. The model is based on the notion of grammatical construction: mapping of word order (surface form) to its meaning. They first transformed the raw sentences by replacing the semantic words (nouns, verbs, etc.) with a unique token 'SW' then the sentences are input to the model sequentially, word by word across time along with the coded meaning (i.e. thematic roles of semantic words) of the input sentence for training. The model learns the thematic roles of all the semantic words in the input sentence during training. During testing, the model predicts the coded meaning of the previously unseen sentences. See chapter 3 for more details about $\theta RARes$ model.

1.2 Motivation and Hypothesis

Like many other traditional NLP systems, Hinaut et al. [20] also treated words as discrete atomic symbols and used localist vector representation of words as an input. Treating each word as a discrete symbol does not provide any relational information to the model which may exist between two words. For example, if words *'pink'* and *'red'* are represented using localist representation with vectors [1,0] and [0,1] respectively, the semantic relationship (i.e. both are colors) between these two words is lost [24]. Although, replacing the semantic words with a 'SW' token makes it possible to train the model on a small corpus as the 'SW' token can be replaced with different semantic words (nouns, verbs, etc.) to form a sentence. Whereas, the 'SW' token in itself does not carry any semantics and thus does not allow the model to take into account the semantics of the words. This makes it

difficult for the model to learn thematic roles for sentences. We discuss the possible limitation of this model later in more detail in Chapter 3.

Motivated by the limitations of localist representation of words and transformation of raw sentences into its abstract form by replacing the semantic words with a 'SW' token described above, this work hypothesizes that the use of distributed word representations, which can capture the syntactic and semantic relationship of words, could improve the performance of the model on TRA task. One such model for learning distributed word embeddings was proposed by Mikolov et al. [33] widely known as the Word2Vec model. Word2Vec model learns high quality, low-dimensional vector representation of words from a large corpus in an unsupervised way. The resulting word vector of this model encodes the semantics of the words. The model learns the word embeddings by taking into account the context (neighboring) words. The obtained word embeddings capture several language regularities and patterns [35, 33] and can be observed by performing the linear operations on the word vectors. For example, *vector('king') - vector('man') + vector('woman')* $\approx$ *vector('queen')*. Unlike other neural network models [38, 13, 52, 36] for obtaining word embeddings, training Word2Vec is computationally cheap and efficient [33]. Training of Word2Vec model and properties of resulting distributed word embeddings will be discussed later in more detail in Chapter 2.

1.3 Proposed Models

This work proposes an end-to-end neural language model for thematic role assignment. The model is named *Word2Vec-θRARes* language model. The Word2Vec-θRARes model is a combination of Word2Vec model and ESN. The Word2Vec model is trained on a general purpose unlabeled dataset (e.g. Wikipedia) prior to the use of the Word2Vec-θRARes model for TRA task. The model receives the raw sentences sequentially and processes a sentence word by word over time. The Word2Vec unit being the first unit in the model receives the input word to generate the distributed word embeddings. The generated word vector by Word2Vec model can then be used by ESN for learning the thematic roles of the input sentences. Note that the proposed model is an extension of the *θRARes* model [20], where, unlike the latter, the raw sentences are not transformed to grammatical forms and word2vec word vectors are used over localist word representation as an input to ESN.

Apart from the Word2Vec-θRARes model, this thesis also proposes a Word2Vec-ESN classifier model which only differs from the former in the way the sentences are processed, and the results are evaluated. Thus in the classifier, the inputs and outputs differs from the Word2Vec-θRARes model. The input feature to this Word2Vec-ESN classifier is the current word (argument) and the predicate with respect to which it is processed. The output units encode the possible role (e.g. Predicate, Agent, Object, Recipient and No Role) of the input argument-predicate pairs, unlike the θRARes model where output units encode the possible thematic roles of all semantic words in the input sentences. For the evaluation

of this Word2Vec-ESN classifier, the metrics proposed for Conll-2004 [10] and CoNLL-2005 [9] SRL task is used. Both Word2Vec-θRARes language model and Word2Vec-ESN classifier model are discussed in more detail in Chapter 4.

There are several neural network based models to obtain the word the word embeddings [38, 13, 52, 36] but a systematic comparison of them on TRA task is beyond the scope of this work. To this date, there is no research conducted with this combination of word2vec model and ESN. This also makes this study novel.

1.4 Outline

This thesis is organized as follows. Chapter 2, describes the basics of Word2Vec model and the Echo State Network model. The chapter also describes in detail the training of the Word2Vec model and properties of resulting word vectors. Also, this chapter describes training and control parameters of ESN. Chapter 3 looks upon related work with the primary focus on neural network $\theta RARes$ language model, its limitations, the motivation and hypothesis for the current work. Subsequently, chapter 4 proposes the Word2Vec-θRARes model and the Word2Vec-ESN classifier. This chapter also describes the implementation and the metrics used to evaluate the performance of the proposed models. Chapter 6 first describes the corpora used and then the experimental setup along with the experiments performed using the proposed models. We will then see the results obtained using the proposed model on the TRA task. In this chapter, we also see the comparison of the results of obtained using the proposed models and the $\theta RARes$ model and analyze them. Finally, in chapter 7 we conclude this study and make the suggestions for possible future work.

Chapter 2

Basics of Word2Vec model and Echo State Network

This chapter establishes the background knowledge required for this research work. The chapter is divided into two section. In the first section, we will see the basics of the Word2Vec model, the flavors in which the models exists, the training of the Word2Vec model and also the properties of the word embeddings obtained by training the model. In the second section, we will have an overview of the Echo State Network and how it is trained.

2.1 Word2Vec model

Word2vec is a neural probabilistic language model proposed by Mikolov et al. [33]. The model is based on the distributional hypothesis which states that the words that appear in the same context share the semantic meaning [24]. The main idea behind the model is to learn the word embeddings, only by its context (nearby words). The model takes a large text data, to generate the distributed embeddings of the words present in the text and also preserve the linear regularities among words. In other words, it maps the words into a continuous vector space where semantically related words are placed close to each other. Earlier the words have been treated as discrete atomic symbols in all traditional NLP systems, where the words are represented in a localist fashion. Localist representation of words does not contain any semantic or syntactic information about the word and thus depriving the NLP systems to utilize this information while processing [4]. Word2vec neural word embeddings overcome this issue and capture the semantics and syntactic information of the word in a computationally-efficient manner [34]. For learning the word embeddings, the Word2Vec model exists in two neural architectures, the Continuous Bag Of Word (CBOW) and Skip-Gram (SG) [34, 33]. Both the models are architecturally the same i.e. both have three layers, the input layer, hidden layer and an output layer, but have different training objectives. The architecture of both CBOW and SG models is shown in figure 2.1 and 2.2 respectively. In the next sections, we give a brief overview of CBOW model and

SG model. However, we used the skip-gram model in our work because it was shown that it produced better word-embeddings as compared to CBOW [33].

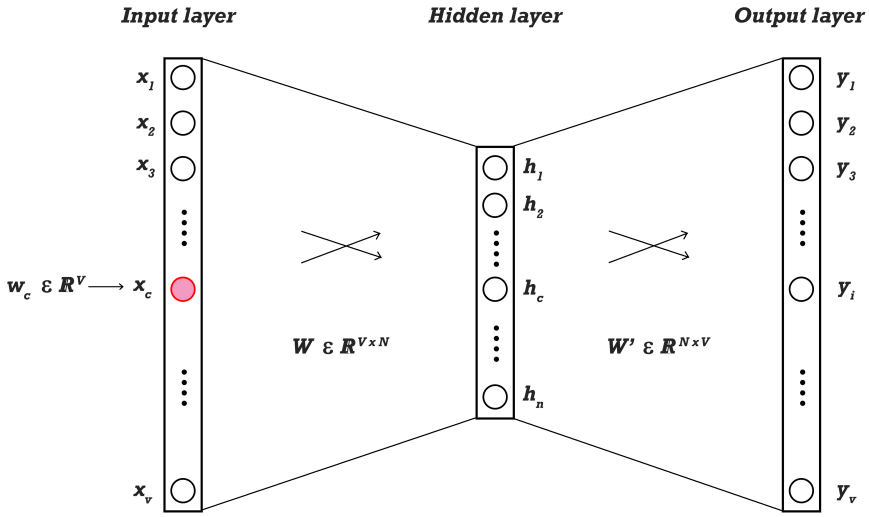

Figure 2.1: **The CBOW model:**In the CBOW model, the objective is to predict the target word based on the words in the context (or neighboring words). The context words are input to the model (in this case only one) using localist representation where only one vector element corresponding to the input word is active (x_c, shown in red). The model outputs the probability for each word in vocabulary, which is maximized for the actual target word during training. Adapted from [44].

2.1.1 CBOW Model

CBOW model is a three layered neural model with the training objective to predict a target word (e.g. Peter) given some context words (John gave the ball to). Figure 2.1 shows the CBOW network architecture with a simplified case of one context word. The model is trained on the dataset having a vocabulary size V in an unsupervised way to achieve the objective. The hidden layer is of size N, the dimensions of the desired word embeddings, and the neurons in both the adjacent layer i.e. input and output layers, are fully connected to the hidden layer neurons. The input to the network is the context word $w_c \in \mathbb{R}^V$ and is represented using localist representation where only one unit x_c at index c will be 1 out of V units $w_c = [x_1, ..., x_V]$ and all other units will be 0 [44]. The activation of the hidden layer is then given by:

$$h = W^T.w_c = W_{(c,:)} = v_c \tag{2.1.1}$$

where $W \in \mathbb{R}^{V \times N}$ is the weight matrix from input to hidden layer and v_c is the vector embedding of the context word w_c. Eqn. 2.1.1 basically copies the c^{th} row of weight matrix W on the hidden layer as the hidden layer activation function is linear.

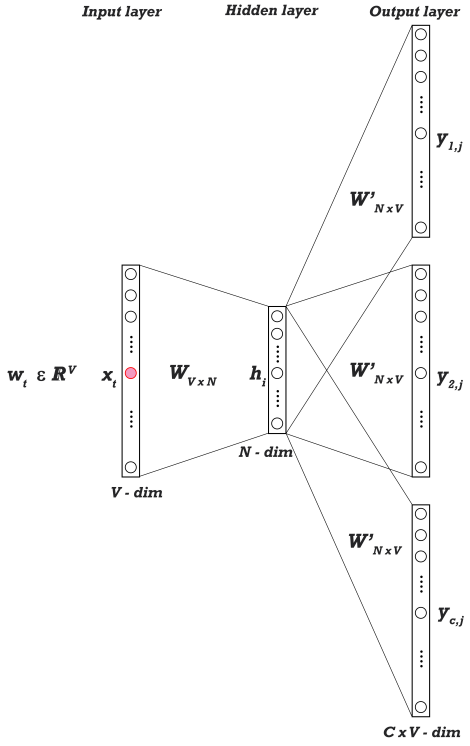

Figure 2.2: **The skip-gram model:** In the skip-gram model, the objective is the reverse of CBOW. It predicts the context words from a target word. The target word is input to the model using localist representation where only one vector element corresponding to the input word is active (x_t, shown in red). The model then maximizes the probability of context words during training. Adapted from [44].

A score $u \in \mathbb{R}^V$ is then calculated for all the target words in the vocabulary, which is essentially the compatibility of a word w_i given the context word w_c.

$$\begin{aligned} u &= W^{'T}.h \\ &= W^{'T}.v_c \end{aligned} \tag{2.1.2}$$

$$u_i = W_i^{'T}.v_c \tag{2.1.3}$$

where u_i gives the score of i^{th} word,w_i, for i = 1, 2...V. $W^{'} \in \mathbb{R}^{N \times V}$ is the weight matrix between hidden and output layer. $W_i^{'}$ is the i^{th} column vector of matrix $W^{'}$. The computed scores are then converted to posterior probabilities by output neurons with softmax activation function [44]. Thus aliasing $W_i^{'}$ as $v_i^{'}$ we get output probabilities as:

$$\begin{aligned} y_i = P(w_i|w_c) &= \frac{\exp(u_i)}{\sum_{i' \in V} \exp(u_{i'})} \\ &= \frac{\exp(v_i^{'T}.v_c)}{\sum_{i' \in V} \exp(v_{i'}^{'T}.v_c)} \end{aligned} \tag{2.1.4}$$

where y_i is the probability of i^{th} word given the context word.

The training objective is then achieved by maximizing the log likelihood of the actual target word (w_t) given the context word (w_c). So the cost functions can be written as:

$$\begin{aligned} J_{ML} &= max \log P(w_t|w_c) \\ &= v_t^{'T}.v_c - \log \sum_{i' \in V} \exp(v_{i'}^{'T}.v_c) \end{aligned} \tag{2.1.5}$$

In the case, when multiple context words are input to the network, the equation 2.1.1 only changes to:

$$h = \frac{1}{K}.(v_{c_1} + v_{c_2} + ... + v_{c_K}) \tag{2.1.6}$$

where k is the size of the context window. This equation averages vector embeddings of all context words [44].

2.1.2 Skip-gram Model

In the Skip-Gram model, training objective is reversed from that of the CBOW model. In other words, the objective is to learn the vector representation of the word that is good in predicting the context words [33]. Thus for a given sequence of words $\{w_1, ...w_V\}$, the objective is to maximize the average log probability.

$$\frac{1}{V} \sum_{t=1}^{V} \sum_{-c \leq j \leq c, j \neq 0} log\ P(w_{t+j}|w_t) \tag{2.1.7}$$

where c is the size of the context window, $P(w_{t+c}|w_t)$ is the probability of the context word w_{t+j} for $-c \leq j \leq c$, given the target word w_t. This is measured using the softmax function as:

$$p(w_{t+j}|w_t) = \frac{\exp(v_{t+j}^{'T}.v_t)}{\sum_{w \in V} \exp(v_w^{'T}.v_t)} \tag{2.1.8}$$

where $v_{t+j}^{'}$ and v_t are the vector representation of word w_{t+j} and w_t respectively.

The objective function is thus optimized using stochastic gradient descent to learn the good word vectors. Calculating the full softmax is computationally expensive as it needs to compute and normalize probability for every other word w

in the vocabulary V for a given input word (w_c for CBOW or w_t for skip-gram) at every training step. Thus negative sampling approach was proposed for learning the word embeddings [33].

Skip-gram with negative sampling

For learning word features full probabilistic models were not required. So, skip-gram negative sampling is used for approximation of word features. This technique treats feature learning as a binary classification (logistics regression) problem [33, 24]. The model is thus trained to distinguish the target word from k imaginary noise words w_{noise}, in the same context. Thus the log probability $p(w_{t+j}|w_t)$ in equation 2.1.8 is now approximated by:

$$p(w_{t+j}|w_t) = log\ \sigma({v'_{t+j}}^T.v_t)\ +\ \sum_{w_{noise}\in N_k} \log\ \sigma({v'_{noise}}^T.v_t) \tag{2.1.9}$$

where $\sigma(x) = 1/(1+\exp(-x))$, and N_k is the set of k noise word compared to the corresponding context word w_{t+j} for $-c \leq j \leq c$.

In order to accelerate the learning and improve the quality of words vectors, the most frequently occurring words in the training corpus can be subsampled [33]. A large training corpus usually contains some words like "a", "the", "in" etc. which occurs million of times. The model benifits less from the most frequent words as compared to the words which occur infrequently. For example, the co-occurrence of "man" with "the" is less important as compared of co-occurrence of "man" with "woman" because the word "the" occurs with most of the words as well [33]. So in order to reduce the effect of most frequent words, each word (w_t) in the vocabulary of the training corpus was discarded with probability:

$$P(w_{w_t}) = 1 - \sqrt{\frac{t}{f(w_t)}} \tag{2.1.10}$$

where $f(w_t)$ is the frequency of word w_t and t is a threshold. Using this formula, the frequencies of the words are preserved while all the words with frequency greater than t are subsample [33].

2.1.3 Properties of Word2Vec embeddings

Although the Word2Vec model is simple in its architecture and easy to train, it produces the word embeddings which surprisingly encodes several linguistic regularities and patterns [35, 33]. Moreover, it is astonishing because the network was not explicitly trained for these linguistic properties (see fig. 2.3 and 2.5). The distributed word embeddings encode the semantic and syntactic properties of the words as a constant vector offset between a pair of words sharing a categorical relationship [33]. For example, the word embeddings "$King - Queen \approx man - woman$", "$apples - apple \approx cars - car$", "$walking - walked \approx swimming - swam$".

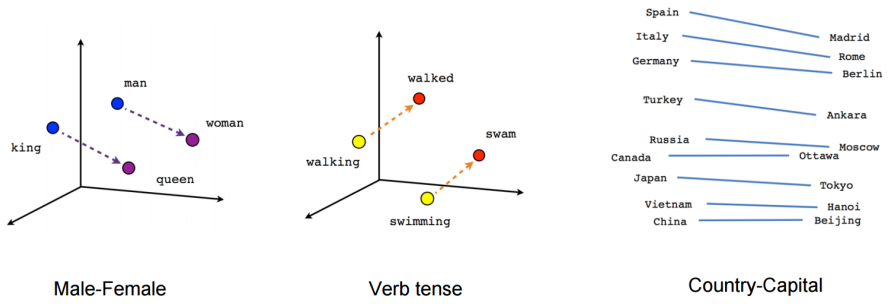

Figure 2.3: **Word2Vec semantic regularities.**

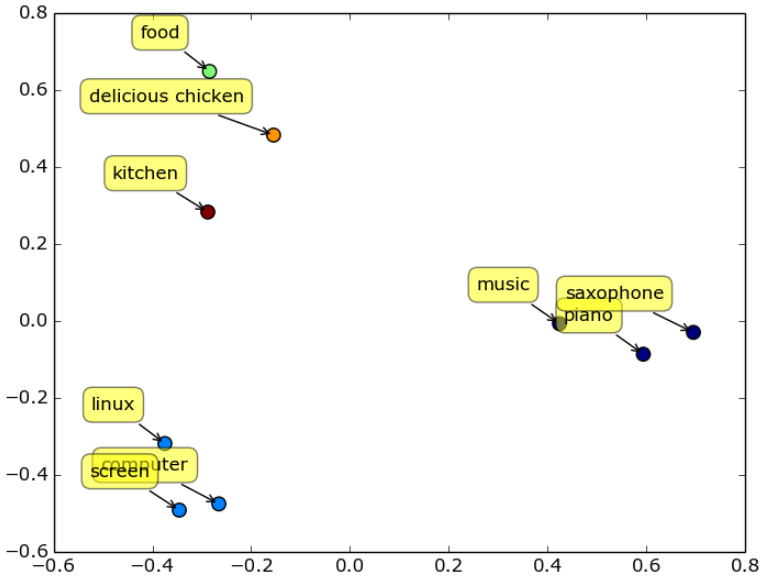

Figure 2.4: **Clustering of words with Word2Vec word embedding:** The figure shows the word clustering property obtained by projecting the word vectors on two dimesional space using PCA. The words vector taken from pretrained word2vec Google News corpus[1]

Another interesting property of distributed word embeddings is that the semantically related words are placed close to each other in the word vector space, thus forming clusters of semantically related words. It was also observed that the word embeddings of similar words in different languages have the same geometrical arrangement in word embedding space of the respective language. Thus it is also possible to learn the linear mapping between different embedding space by vector rotation and scaling [35]. Several other regularities can also be captured by performing the basic linear operation on word-embeddings [33].

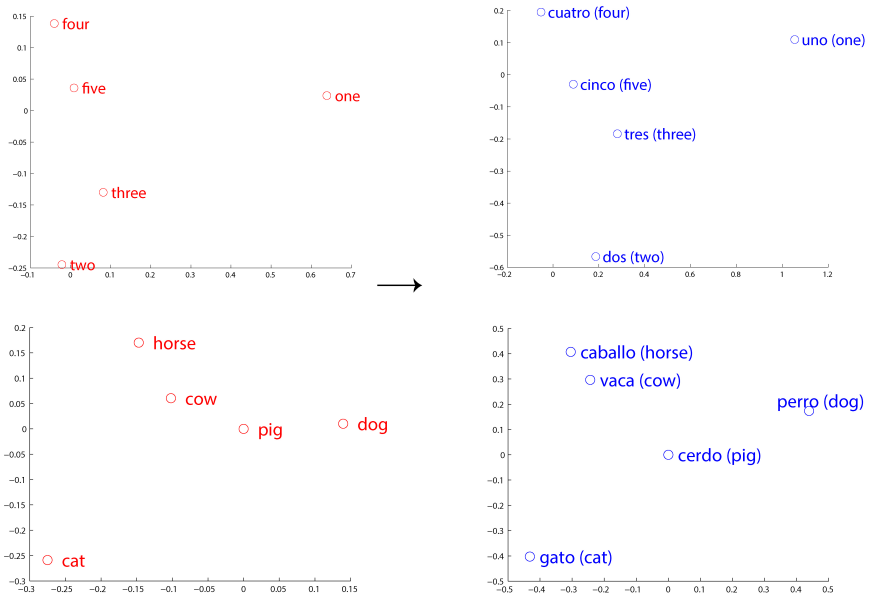

Figure 2.5: **Word2Vec language translation property.**

2.2 Echo State Network (ESN)

Echo State Network (ESN) is a network with a new viewpoint on Recurrent Neural Network (RNN). It is a discrete time continuous state recurrent neural network introduced by Jaeger [27] and is believed to closely resemble the learning mechanism in biological brains. ESN is found to be computationally simple and inexpensive to process the temporal or sequential data. The main idea of ESN is to operate the random, large, and crucially, fixed RNN with the input signal. The non-linear response generated by each neuron of the RNN is then collectively combined with the desired output signal using regression to learn the output weights [26, 27, 25]. It is worth noting that the ESN are not trained using error Back Propagation (BP) [45] but, instead the linear regression is used. The reason for this choice is that, using BP in RNN makes the training of RNN non-convergin, slow, and computationally expensive [16]. Whereas training using linear regression is faster and computationally inexpensive [32].

2.2.1 ESN Architecture

ESN is a variant of RNN to train them efficiently. Figure 2.6 shows the basic architecture of an ESN. In the standard RNN, all the weights are required to be tuned, even though it was shown that RNN works well enough even without full adaptation of weights [32]. The classical ESN mainly contains three layers, input layer, the hidden layer (also known as the reservoir) and the readout layer. The input layer is fully connected to the hidden layer, and both the hidden layer and the input layer are connected to the output layer. The output layer is fully connected

back to the hidden layer. However, the connections from input to output layer and output layer to hidden layer are optional and depends on the task.

The weights from input to reservoir (i.e. W^{in}) and from reservoir to reservoir (i.e. W^{res}) are sparsely (i.e. most of the elements in these matrices will be zero) and randomly initialized. The sparsity of W^{in} and W^{res} is however not a necceray condition but Jaeger [27] recommended the sparse initialization of these weights. The non-zero element in the sparse input weight matrix, W^{in}, and the reservoir weight matrix, W^{res}, are generated from uniform or normal distribution. W^{in} and W^{in} weights remains untrained during training whereas the weights from the reservoir to output layer (i.e. W^{out}) are the only weights learned during supervised training [27, 32]. For the ESN approach to work, the reservoir should possess the ***Echo State Property (ESP)***: if a long input sequence is given to the reservoir the reservoir will end up in the same state irrespective of the initial reservoir state. In other words, the reservoir states 'echoes' the input sequence and the effect of the previous reservoir state and the previous input on the future reservoir states should vanish gradually [32, 26].

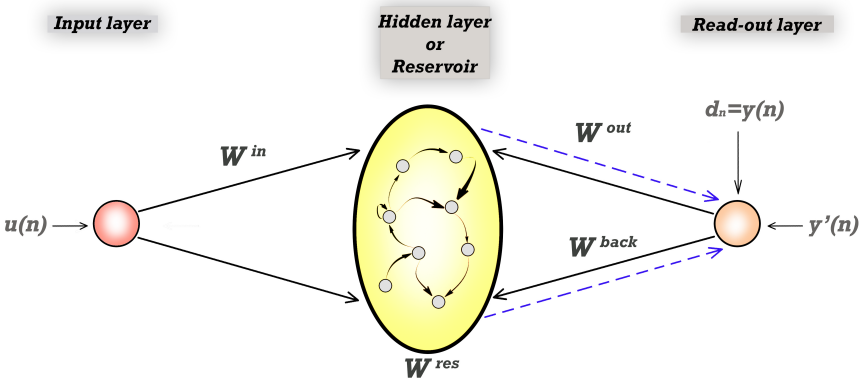

Figure 2.6: **Architecture of classical ESN:** The reservoir is the recurrent neural network with N_x units and initialized with sparse and random connections. The reservoir is provided with an input sequence $u(n)$ from the input layer and the teacher signal $y(n)$ from the output layer during training. The input to reservoir weights (W^{in}), output to reservoir weights (W^{back}, optional and depends on task) and reservoir to reservoir weight (W^{res}) are randomly initialized and remanins static during learning. The output weight from the reservoir to output unit are the only weights learned by the network during training. Adapted from [32]

To ensure the echo state property in ESN, firstly, the reservoir weights matrix W^{res} and the input weights matrix W^{in} are often generated sparsely (not necessarily) and randomly from a normal or uniform distribution [32]. The input weight matrix is, however, a bit denser than the reservoir weight matrix. The sparsely generated random reservoir weight matrix W^{res} is often scaled such that its spectral radius, $\rho(W^{res})$, i.e. largest absolute eigenvalue, is less than one [25]. However $\rho(W^{res}) < 1$ is not a necessary condition for ESN to have the ESP and can be achieved even when $\rho(W^{res}) > 1$ [27, 32, 26]. Intuitively, the spectral radius is

a crude measure of the amount of memory the reservoir can hold. The smaller values are used when short memory is required, and the larger values for a longer memory[32].

To scale the randomly generated W^{res} matrix, it is first divided by its spectral radius and then multiplied with the desired spectral radius [26].

$$W^{res}_{new} = \gamma \frac{W^{res}}{\rho(W^{res})} \tag{2.2.1}$$

where W^{res}_{new} is the scaled reservoir weight matrix, γ is the desired spectral radius and $\rho(W^{res})$ is the spectral radius of the randomly generated reservoir Matrix W^{res}.

The input weights (W^{in}) are also scaled to regulate the non-linearity in reservoir activations. A very high input scaling let the reservoir behave in a highly non-linear manner (because of the 'tanh' activation function) whereas a very small input scaling is used wherever linearity is required in a task [32].

2.2.2 Training ESN

ESNs are mostly applied for supervised machine-learning tasks where the temporal or sequential aspect of the data is to be modeled. Before training, the reservoir of size N_x, generally containing leaky-integrated discrete-time continuous-value neurons with *tanh* activation function are generated. The reservoir of any computationally affordable size can be used. The bigger the reservoir size, the more the input signal gets non-linearly expanded and easier it will be to find the linear combination with the desired output signal [32, 26]. The bigger reservoir size is however not always a best choice. With the big reservoir size comes the risk of over-fitting. Thus, it is also important to use proper regularization methods to avoid over-fitting. After generating the reservoir, the reservoir weights, W^{res}, the input weights, W^{in}, and the feedback weights, W^{back}, are randomly initialized.

The training objective of the ESN approach is to learn a model which outputs y', such that it is as close as possible to the target output y by minimizing the some error measure, $E(y', y)$. The learned model should also be able to generalize well on the data not used for training. Root Mean Square Error (RMSE) is typically chosen as an error measure E. Thus during training, the given input signal $u(n) \in \mathbb{R}^{N_u}$ and the corresponding teacher signal $y(n) \in \mathbb{R}^{N_y}$ are input to the reservoir at every time-step 'n'. Here n = 1, 2..., T is the discrete time step for sequence of length T. The reservoir then generates a sequence $x(n) \in \mathbb{R}^{N_x}$ reservoir states which is a non-linear high dimensional expansion of the input signal $u(n)$ [26]. The reservoir activation and reservoir state update is computed using following recursive equations:

$$x'(n) = tanh\ (W^{res}x(n-1) + W^{in}.u(n) + W^{back}.y(n-1)) \tag{2.2.2}$$

$$x(n) = (1-\alpha)x(n-1) + \alpha x'(n) \tag{2.2.3}$$

where $x(n)$ is the vector of reservoir neuron's activations and $x'(n)$ is its update at time step n and *'tanh'* is the reservoir neuron activation function. $W^{in} \in \mathbb{R}^{N_x \times N_u}$

and $W^{res} \in \mathbb{R}^{N_x \times N_x}$ are input weights and reservoir weights matrices respectively. $W^{back} \in \mathbb{R}^{N_x \times N_y}$ is the optional output to reservoir matrix [2] and $\alpha \in (0, 1]$ is the leaking rate of neurons.

The leaking rate, α, regulates the speed of reservoir update dynamics in discrete time. A smaller value of α induces slow reservoir dynamics thus ensuring the long short-term memory in ESN [32, 29]. The reservoir activation states are accumulated at every time step for regression with the teacher output. The linear readout weights are then learned using equations:

$$y'(n) = W^{out}x(n) \tag{2.2.4}$$

where $y'(n) \in \mathbb{R}^{N_y}$ is the output of the network and $W^{out} \in \mathbb{R}^{N_y \times N_x}$ is the output weight matrix.

Writing the equation 2.2.4 in matrix form, the output weights W^{out} are then learned using the following equation:

$$Y' = W^{out}X \tag{2.2.5}$$

$$W^{out} = Y'X^T(XX^T + \beta I)^{-1} \tag{2.2.6}$$

where β is the regularization coefficient parameter of ridge regression and I is the Identity matrix.

Training procedure of ESN has only a few global parameters which are to be optimized: the reservoir size N_x, spectral radius of W^{res}, input scaling of W^{in}, leak rate α and the ridge parameter β. All these parameters can only be optimized by trial and error method and depend heavily on the task under consideration [32]. Usually, a grid search is applied to explore the best parameter combination.

[2] this weight matrix is not used in our model implementation

Chapter 3

Related Work: $\theta RARes$ model and its limitation

Humans have a remarkable ability of perceiving and comprehending one or more languages. But how do they know what does a sequence of symbols means? In other words, how do they link a series of words to its meaning? With this research question, Hinaut et al. [20] proposed a neuro-inspired model, $\theta RARes$, to process the sentence across time without having to know the semantics of the words. The model uses reservoir computing approach namely Echo State Network (ESN) and has been successfully implemented on iCub humanoid robot platform [21] for the Thematic Role Assignment (TRA) task. This was the first time when ESN was used for TRA task. The model is based on the cue competition hypothesis of Bates et al. [6] which states that closed class words (e.g. prepositions, articles, determiners, pronouns, etc.), the order of words and prosody in a sentence encodes the grammatical structure. In order to map the sentence structure to its meaning, the model also relies on the assumption made by Dominey et al. [15], according to which the meaning of the sentence depends on the closed class words and markers. The experiments performed using θRARes model for TRA task, showed the results toward modeling of language acquisition in brain [23, 20]

3.1 Overview of θRARes Model

θRARes Model is basically an ESN used to learn and predict thematic roles of the input sentences. The model is based on the notion of grammatical construction [20]. The grammatical construction of a sentence is defined as the mapping of the surface form (word order) of the sentence to its meaning [19]. Figure 3.1 represents the characterization of thematic role assignment in grammatical construction. The model does process the raw sentences but instead use the abstract form of the sentences. The abstraction marks each word of a sentence into two kinds of symbols: Functional Words (FWs) and Semantic Words (SWs). Semantic words are the open class words. An open class is the one which accepts new words and usually includes nouns, verbs, adjectives, adverbs, etc. [2]. Functional Words

are closed class words or grammatical words. A closed class is the one which contains a limited number of words. In a closed class new words are rarely added and usually includes determiners, prepositions, articles, pronouns and verb inflections like -ed,-ing or -s, etc. [2].

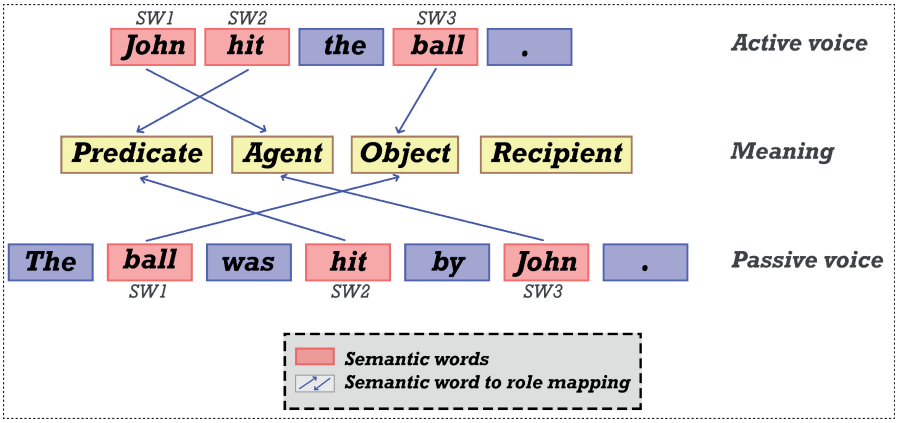

Figure 3.1: **Schematic characterization of thematic roles in grammatical construction:** The figure shows the mapping (shown with arrows) of semantic words (shown in red) to their thematic roles for an active and passive sentence. Figure adapted from [20]

Figure 3.2 shows the functional organization of the model. As the closed class contains a limited number of words and new words are rarely added to it, a set of closed class words is defined for the TRA task. Before giving a sentence as an input to ESN, all the semantic words (not in closed class words set) are replaced with a unique token, 'SW', and pushed to FIFO memory stack (see fig. 3.2). The functional words are left unchanged as the functioning of model completely depends on the order of closed class words [20]. This transformation of sentences to grammatical form by replacing the semantic words with 'SW' tokens allows the θRARes model to be trained on a small dataset. The reason is that one grammatical form can represent several sentences just by replacing different nouns, verbs, etc. with the 'SW' token.

To train the θRARes model, the transformed sentences are presented to the model sequentially, word-by-word across time. Considering each word as a discrete atomic symbol, the words in a sentence are encoded using localist vector representation, where all the elements are zero except for the current input word (see table 3.1). The localist word vectors are projected on the input layer of ESN, and the coded meaning of the input sentence is teacher-forced on the output layer of ESN. The model learns the thematic roles of all semantic words by learning reservoir to readout weights of ESN. During the testing, the model predicts the coded meaning of the test sentences. The output activation of the model is decoded to its meaning by thresholding the activation at last time step. For each 'SW' the role that has the highest activation is considered as the role of 'SW'. Each semantic word in the FIFO memory stack is then mapped to the corresponding role to obtain the coded meaning of a sentence.

Table 3.1: Transformation of a sentence to an abstract form by replacing semantic words with 'SW' token and localist vector representation of words for a sentence.

Original words	Transformed words	Localist vectors					
put	SW	0	1	0	1	0	1
the	the	0	0	1	0	0	0
ball	SW	0	1	0	1	0	1
on	on	0	0	0	0	1	0
the	the	0	0	1	0	0	0
box	SW	0	1	0	1	0	1

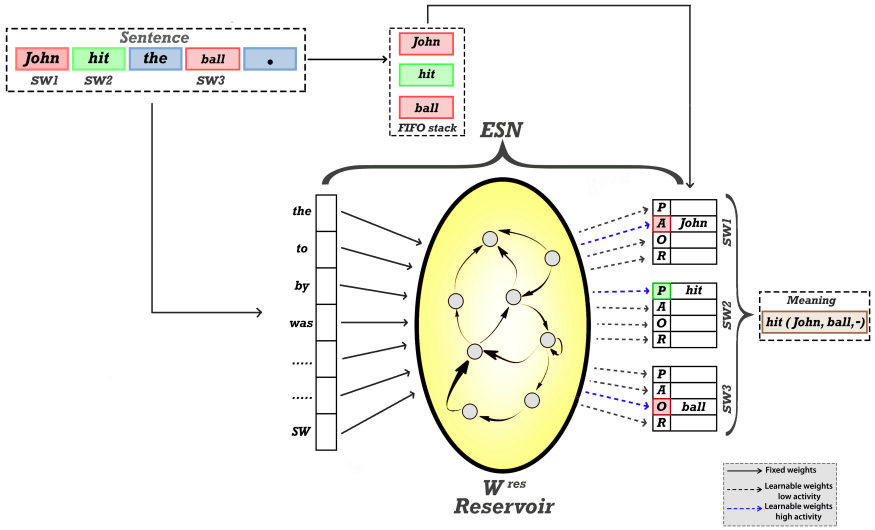

Figure 3.2: **Functional organization of θRARes model.**

3.1.1 Limitations of θRARes model

Transforming a sentence into its abstract representation (explained above) and using the localist representation of each word in a sentence as an input to an ESN does not allow the network to leverage the information retrieved from semantically related words [24]. The limitations of such an input representation mainly occur with the ambiguous sentences. A sentence is said to be ambiguous if it has the same grammatical form but the different coded meaning and actual meaning (see fig. 3.3). The following two example illustrates the limitations of using localist words representation and grammatical transformation of sentences.

Example 1: Ambiguous Sentences

1. take (SW-1) the blue (SW-2) box (SW-3)
2. take (SW-1) the left (SW-2) box (SW-3)
3. throw(SW-1) the green(SW-2) box(SW-3)

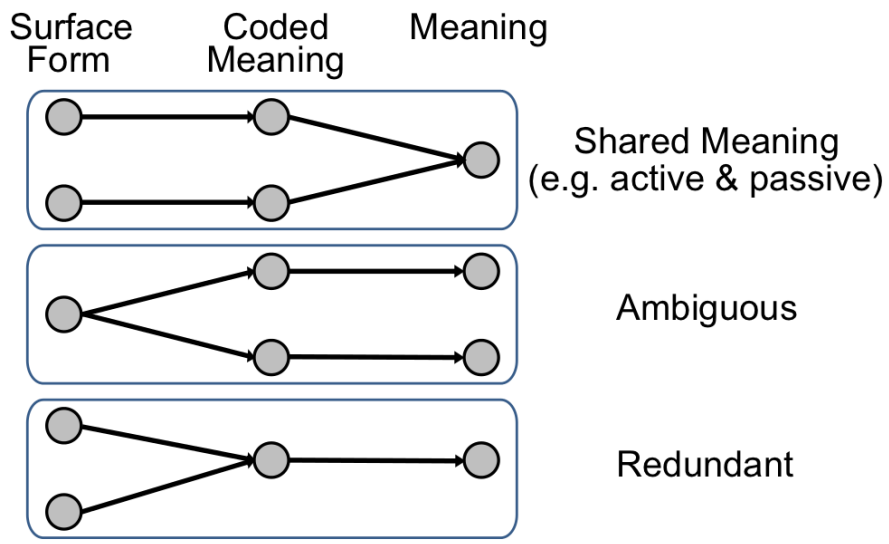

Figure 3.3: **Relation between surface form and meanings:** sentence structure to meaning mapping exist in three categories. Active and passive sentence like "John chased the dog" and "The dog was chased by John" have the different surface form but share the same meaning i.e. chased(John, dog). Also, the coded meaning of both these sentences are different i.e. SW-3 ("dog") in the active form is the 'Object' whereas in passive form SW3 ("John") it is an 'Agent'. Ambiguous sentences have the same surface form, but different meaning and coded meaning whereas redundant sentences have different surface form but the same meaning and coded meaning. Adapted from [55]

All the three sentences having the same surface form when transformed to abstract form by replacing the semantic words with 'SW' token i.e. SW the SW the SW SW.

Training θRARes model with sentence 1: In the sentence 1 there are three semantic words namely SW-1: take, SW-2: blue, SW-3: box. During the training, the sentence is input to the model from left to right word by word at each time step. The teacher output (thematic roles) of each semantic word is also teacher-forced as described below where, A: Action, O: Object, C: Color, I: Indicator. During the training ESN learns that the second semantic word represents a *'Color'*.

SW-1			
A	O	C	I
take			

SW-2			
A	O	***C***	I
		blue	

SW-3			
A	***O***	C	I
	box		

Training θRARes model with sentence 2: In the sentence 2, we have three semantic words; SW-1: take, SW-2: left, SW-3: ball. Notice that the second semantic word (SW-2) in both the sentences (1 and 2) differs. Now suppose we

train this sentence with a teacher output as follows where, symbols A, O, C and I, have the same meaning as described above.

SW-1			
A	O	C	I
take			

SW-2			
A	O	C	***I***
			left

SW-3			
A	***O***	C	I
	box		

As both the sentences used for training have the same surface form, the ESN learning from the sentence 1 may get disrupted after training with sentence 2. This is because, the second semantic word (SW-2) in the sentence 2 is trained to be an *'Indicator'* whereas it was trained for role *'Color'* in the sentence 1. Such kind of ambiguous sentences having the same surface form but different coded meaning (roles assignment) and actual meaning, drops the learning ability of the model. We believe that this problem is mainly because each input words are treated as discrete atomic symbols, which does not carry any semantic information about the input words.

Testing θRARes model with sentence 3: Now, when we use the sentence 3 for testing, the network may suggests two pseudo probabilities for the second semantic word, *green*, i.e. being an *'Indicator'* and a *'Color'* as shown below. It becomes difficult for the model to resolve this ambiguity and assign the appropriate role. Although, the word *green* in the test sentence 3 and the word *blue* in the training sentence 1 are semantically related i.e. both are colors. The model cannot utilize this information as the words are input to the model in localist fashion for training. Thus by transforming a sentence to abstract grammatical form and using localist vector representation, the model is deprived of the semantic information carried out by a word in a language [24].

SW-2			
A	O	***C(0.5)***	***I(0.5)***
		green	***green***

Similary, the following three sentences also have the same suface form i.e. The SW is SW to SW.

(i) The chicken (SW-1) is cooked (SW-2) to eat (SW-3)
(ii) The ball (SW-1) is given (SW-2) to John (SW-3)
(iii) The book (SW-1) is taken (SW-2) to John (SW-3)

Clearly, we can see that the third semantic word (SW-3) is a *'Predicate'* in sentence (i) and *'Noun'* in sentence (ii). Thus if network is trained on sentences (i) and (ii) and tested on sentence (iii) the network may not be able to resolve the ambiguity for semantic word SW-3, which possibly leads to wrong labelling of this word.

Example 2: Ambiguous and Polysemous Sentences

1. John (SW-1) books (SW-2) the ticket (SW-3) to London (SW-4)
2. John (SW-1) read (SW-2) the books (SW-3) to learn (SW-4)

Both the above sentences share the same surface form i.e. SW SW the SW to SW.

Training θRARes model on sentence 1: Training the model on the first sentence, the fourth semantic word, *London*, is trained for the role *'Location'*.

Testing θRARes model on sentence 2: While testing the model on the second sentence, the fourth semantic word, *learn*, will be assigned the role *'Location'* (as network was only trained for this) whereas it is actually an *'Action'*. The reason for this mislabelling lies in the approach used to train the θRARes model. Replacing the semantic words with 'SW' token and treating each word in the sentence as a discrete atomic symbol removes the semantic and syntactic information carried out by a word in a language. Hence the model could not exploit the semantic information carried by a word while learning the mapping between a word and the corresponding role.

Issue with Polysemous words: In the first sentence, the second semantic word, *books*, is the *'Predicate'* and describe an action of making a reservation. Whereas the same word, *books*, in the second sentence is an *'Object'*, and represents a book which we read. Although both words are same, they represent different meanings depending on the context. These kinds of words with various meaning are also known as ***Polysemous words***. Semantic ambiguities because of polysemous words are hard to resolve with localist vector representation where each word is treated as an unique identifier. To resolve such kind of semantic ambiguities, distributed embedding of words can be useful as they are learned using the context words and carries semantic and syntactic information about the words (e.g. Word2Vec word vectors) [33, 34].

3.1.2 Research Hypothesis

Transformation of a sentence to its abstract grammatical form and using the localist word vectors as an input to the θRARes model has produced the promising results for the TRA task [20, 23, 21]. However, the behavior of the model with the distributed word embeddings was left unexplored. Thus inspired by the limitations of θRARes model described above, this research work hypothesizes that using the distributed word embeddings (e.g. Word2Vec word embeddings) could improve the performance on TRA task. The rationale behind using distributed word embeddings to represent the word is the fact that they capture and encode semantic and syntactic information carried out by words [33, 50]. Another advantage of using distributed word embedding observed over localist vector representation is

that the multidimensional distributed vector representation of semantically related words remains close in the neighborhood of words embedding space (see fig. 2.3) [33]. This clustering of semantically related words is however not possible when the words are represented using localist words vectors. Thus the use of distributed word embeddings could allow the θRARes model to learn these dynamics and avoid the disruption caused by semantically unrelated words (as explained in Example 1) in the sentences with the same grammatical constructions.

Consider using distributed embeddings of words for both the sentences in Example 2. The distributed embedding of the word, *London* (sentence 1, Example 2), encodes that it represents a *'Location'* along with several other semantic information [34]. Similarly, the distributed vector representation of the word, *learn'* (sentence 2, Example 2), also encodes that it is a *'Predicate'* along with other semantic information. When these distributed embeddings are used as an input to train the model, the learning of the model may not get disrupted by the sentences having the same surface form, but different semantic words. Hence, thematic roles could be assigned to the words more accurately although the sentence has the same surface form.

Chapter 4

Approaching Word2Vec-θRARes Language Model

So far we have seen the basisc of Word2Vec model and Echo State Network. We have also seen the functioning and possible limitation of θRARes model. In this chapter, we extend the θRARes model and propose the Word2Vec-θRARes neural language model and also a Word2Vec-ESN classifier to validate the research hypothesis.

4.1 Word2Vec-θRARes Language Model

This research proposes a Word2Vec-θRARes language model for TRA task. The proposed model is inspired from the θRARes model proposed by Hinaut et al. [20] for TRA task. Figure 4.1 shows the architecture of Word2Vec-θRARes model. Word2Vec-θRARes model is the combination of Word2Vec model and Echo State Network (ESN). The Word2Vec model is responsible for generating the distributed embeddings of the words in the input sentences. The generated word embeddings can then used as an input to ESN, which further processes these word embeddings to learn and predict the thematic roles of all semantic words in the input sentences.

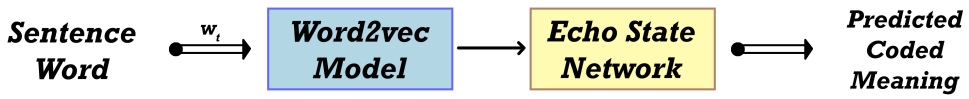

Figure 4.1: **Architecture of Word2Vec-θRARes model:** The model, takes the words of a sentence as an input across time. Word2Vec model generates the distributed vector representation of the input word. The generated word vector is then used by ESN for further processing and learns to predict the thematic roles of the input sentence.

4.1.1 Model initialization

Before using the Word2Vec-θRARes model for TRA task, the Word2Vec model is trained using skip-gram negative sampling [33] approach on a general purpose dataset (e.g. Wikipedia) and the domain specific dataset. During the training of Word2Vec model, the low dimensional distributed embeddings for each word in the corpus vocabulary is learned (see section 5.2.1).

To initialize the ESN, firstly, a reservoir of size N_x composed of leaky integrator neurons and *tanh* activation function is created. The input-to-hidden weights (W^{in}) and hidden to hidden weights (W^{res}) are then generated sparsely and randomly from a Gaussian distribution with mean 0 and variance 1. These weights once initialized are fixed and remains unchanged during training [25, 32]. To generate the weights sparsely, a fixed fan-out of F_{hh} and F_{ih} is chosen for hidden-to-hidden and input-to-hidden connections respectively. In other words, each reservoir neuron is connected to F_{hh} other reservoir neurons and each input neurons are connected to only F_{ih} reservoir neurons.

4.1.2 Training model

Word2Vec-θRARes language model treats the TRA task as a prediction problem. The objective of this model is to learn and predict the thematic roles of all semantic words in the input sentence. To evaluate the performance of this model meaning error and sentence error metrics are used (see section 4.1.3). The same evaluation metrics was used to evaluate θRARes model [20]. Thus it enable us to compare the performance of both Word2Vec-θRARes and θRARes model.

Figure 4.2), shows the neural comprehension of Word2Vec-θRARes model for thematic role assignment. Like θRARes model, a set of closed class words is created prior to training of model. During the training, the sentences are presented to the model one at a time, word-by-word across time. Before presenting a sentence to the model, all the semantic words (not in closed class set) in the sentence are identified and placed in FIFO memory stack (see fig. 4.2). This memory stack will be used later to decode the output coded meaning of the semantic words to the meaning of input sentence. The readout layer of the model is also teacher-forced with the coded meaning of the input sentence. The size of readout layer thus depends on the maximum number of semantic words a sentence can have in the corpus. A semantic word in a sentence can have one of the four possible roles: Predicate (P), Agent (A), Object (O), Recipient (R). For example, if the sentences in the corpus have a maximum of N_{sw} semantic words then the readout layer size would be $N_{sw} \times 4$ neurons; where each output neuron encodes the thematic role of a semantic word. The model could also take the topologically modified but equivalent coded meaning of the sentences (see fig4.3), where the role of each noun in a sentence is represented with respect to a verb [20]. With the topologically modified coded meaning, the readout units contains $N_{noun} \times N_{verb}$, where N_{noun} and N_{verb} are respectively the maximum number of nouns and verbs any sentence could have in the corpus. The output neurons have an activation of 1 if the corresponding

thematic roles are present for the sentence, -1 otherwise.

The Word2Vec model receives the words from an input sentence over time and generates a word vector of E_v dimensions which is then used as an input for the ESN. The input layer of ESN uses this word vector as input features for learning and predicting the thematic roles of the semantic words in the sentences. Thus the size of the input layer is same as the dimensionality of word vector i.e. E_v. The output ofthe reservoir is accumulated for each time step during the presentation of a sentence. The accumulated reservoir states are then linearly combined with readout activations (coded meaning of the input sentence) to learn the reservoir-to-readout (W^{out}) weights using linear or ridge regression. The reservoir states of ESN are reset before the presentation of the consequent sentence. Word2Vec-θRARes model can be operated in two learning modes so that it learns to extract the coded meaning of the semantic words in the sentences.

1. **Sentence Continuous Learning (SCL)**: In this learning mode, learning takes place from the beginning of the sentence. In other words, the coded meaning of the input sentence is made available to model from the onset of the first word of the sentence. Thus, the regression is applied with the teacher roles from the onset of the first word in the sentence [20].
2. **Sentence Final Learning (SFL)**: In this mode, the learning takes places only at the end of the sentence [20]. Hence, the teacher labels are only provided to the network at the end of sentence i.e. from the last word of the sentence.

Decoding Output: As described earlier, the coded meaning of a sentence is defined as the description of thematic roles for all the semantic words in the sentence. As the readout neurons of model codes the thematic roles for all semantic words, the output activations produced by the model while testing a sentence are thresholded at 0 for every semantic words. The maximum of all activation between four possible roles (i.e. Predicate, Agent, Object, Recipient) is taken as the coded meaning of a semantic word [20]. A semantic word is said to have an incorrect coded meaning if the winning readout neuron is not the true role of the semantic word. If there is no activation above the threshold for a semantic word, then this semantic word is considered to have no coded meaning [20]. Combining the coded meaning of semantic words, forms the coded meaning of the sentence. The coded meaning of the sentence can then be decoded to the actual meaning by mapping the coded meaning to the semantic words from the FIFO memory stack(see fig 4.2).

4.1.3 Evaluation Metrics

To evaluate the performance of the Word2Vec-θRARes language model, the same metrics that was used by Hianut et al. [20] to evaluate the θRARes model was

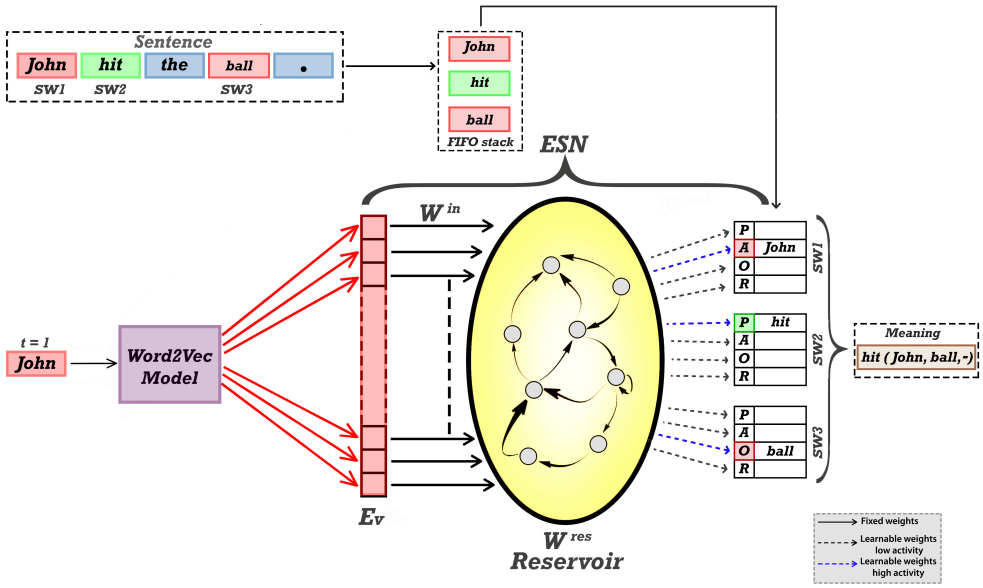

Figure 4.2: **Word2Vec-θRARes language model:** The figure shows the processing of a sentence by the model Word2Vec-ESN classifier at time step 1. Nouns and verbs (specified in red and green respectively) are stored in a memory stack for interpreting the coded meaning. The word *'John'* is input to word2vec model which generate a word vector of E_v dimensions. The output vector is then input to ESN for further processing. During training, the readout units are teacher-forced with the coded meaning of the input sentence. During testing, the readout units generate the activations which code the predicted coded meaning of input sentence. The meaning: hit(John, ball, -) is decoded from coded meaning by mapping the thematic roles with semantic words from memory stack. Adapted from [20]

employed. The metrics compute the Meaning Error (ME) and the Sentence Error (SE). Meaning error is the percentage of semantic words with incorrect coded meaning, whereas the sentence error is the percentage of sentences having at least one semantic word with incorrect coded meaning. Both the error measures are related, but there is no strict correlation between them [20]. Sentence error is a stricter measure than meaning error to evaluate model performance because a meaning error of 5% cannot be used to estimate the sentence error, as this 5% incorrect words, can be from just one sentence or several sentences.

To evaluate the performance of Word2Vec-θRARes model, not all the readout neurons are considered i.e. if a sentence has only three semantic words then only readout neurons corresponding to these three semantic words are analyzed [20] and remaining coded meaning of remaining neurons are ignored. It is important to know that if there are more than one verb, in a sentence then each semantic word can have a possible role with respect to each verb.

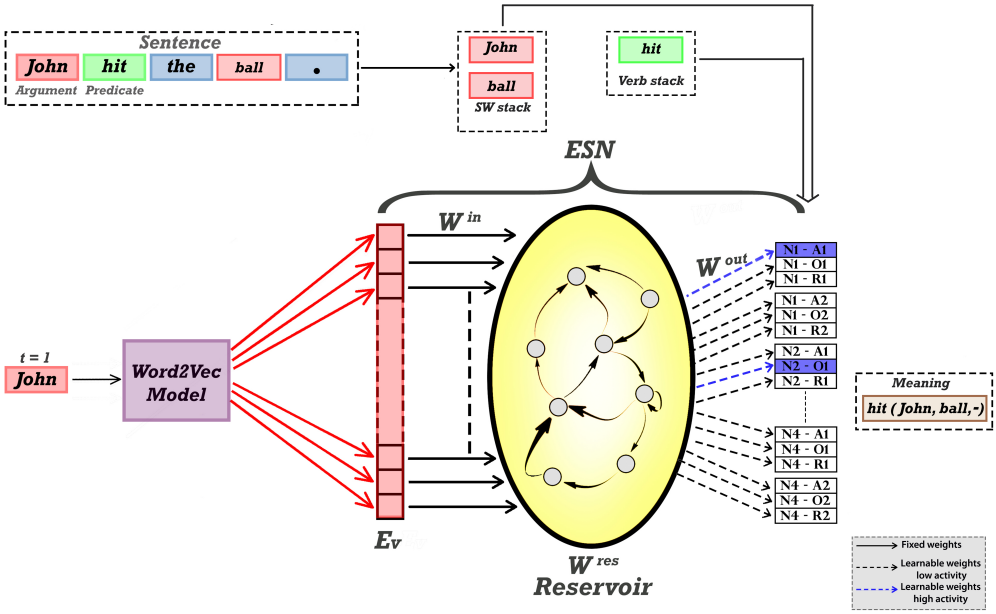

Figure 4.3: **Word2Vec-θRARes language model with toplogically modified output coding:**The figure shows the processing of a sentence by the model Word2Vec-ESN classifier at time step 1. Nouns and verbs (specified in red and green respectively) are stored in a repective memory stack for interpreting the coded meaning later. The word *'John'* is input to word2vec model which generate a word vector of E_v dimensions. The output word vector is then input to ESN for further processing. During training, the readout units are teacher-forced with the coded meaning of the input sentence. The coded meaning "N1-A1" represents that Noun-1 ('John') is agent of Verb-1 ('hit'). During testing, the readout units generate the activations which code the predicted coded meaning of input sentence. The meaning: hit(John, ball, -) is decoded from coded meaning by mapping the thematic roles with nouns and verbs from memory stack. Adapted from [20]

4.2 Word2Vec-ESN classifier

To ensure the objectivity of our findings apart from Word2Vec-ESN model, this research work also propose a Word2Vec-ESN classifier. Figure 4.4 illustrates the functional organisation of Word2Vec-ESN classifier for TRA task. Although the model Word2Vec-ESN classifier is architecturally (see fig. 4.1) similar to Word2Vec-θRARes model, but varies in training objective and the way the sentences are processed. It treats the TRA task as a classification problem. The training objective of Word2Vec-ESN classifier is to classify the words of the input sentences to one of the roles namely Predicate (P), Agent (A), Object (O), Recipient (R) and No-role (XX) with respect to the verbs and maximize the classification scores for each role (i.e. F1-score, Precision, and Recall- see section-4.2.3).

Two input features play a major role in Word2Vec-ESN classifier: argument and predicate. An argument describe the current word being processed and the

predicate describes the verb with respect to which an argument is processed [55]. So, if there are N_v verbs in a sentence, then the same sentence is processed N_v times. Each argument then takes a unique role for an argument-predicate pair. For example in the following sentence there are two predicates namely *'chased'* and *'ate'*. Thus this sentence will be processed twice, and each argument will take a role for an argument-predicate pair [55].

Arguments →	the	dog	that	chased	the	cat	ate	the	rat
Predicate('chased')	XX	A	XX	P	XX	O	XX	XX	XX
Predicate('ate')	XX	A	XX	XX	XX	XX	P	XX	O

It is worth noticing that the classifier takes in input, the predicate with respect to which an argument is processed. To retrieve the predicates from a sentence, the Word2Vec-ESN classifier has to depend on syntactic parser e.g. Charniak parser. This limits the classifier from being an end-to-end system for TRA task.

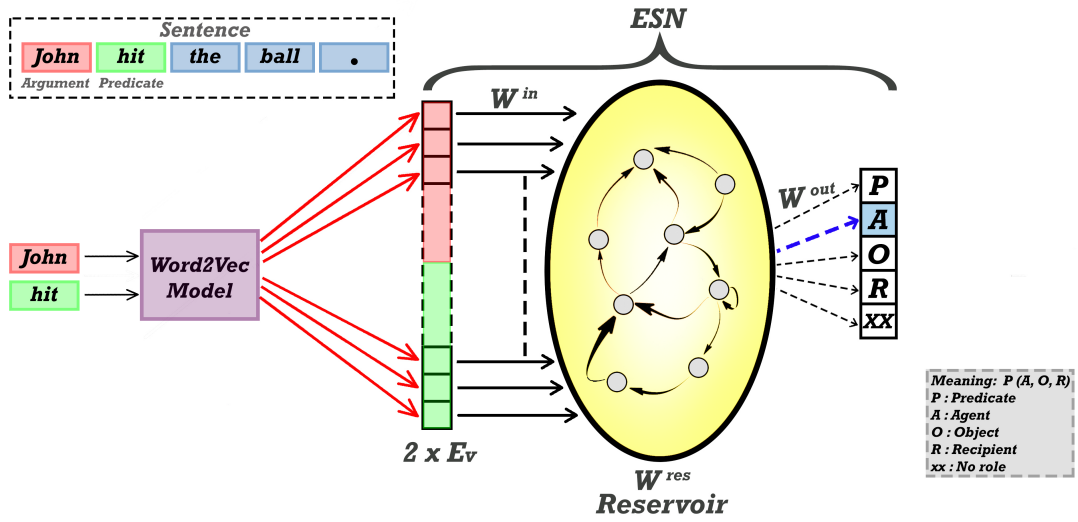

Figure 4.4: **Functional organisation of Word2Vec-ESN classifier for TRA task:** The figure shows the processing of a sentence in Word2Vec-ESN classifier at time step 1. At any instant of time, an argument (current word, marked in orange) and predicate (marked in green) is input to the model. Word2Vec model generates the word vectors of E_v dimensions which are then concatenated to form a $2 \times E_v$ dimensions (shown in red and green color). ESN takes the resultant vector for further processing. During the training, the readout neurons are presented with the role of input word-verb pair (e.g. 'A' for Agent). The read-out weights (shown in dashed line) are learned during training. While testing, the readout unit codes the role of the semantic words, which are then accumulated and decoded to form the meaning *hit(John,ball,–)* of the input sentence at the end. Inspired from [20]

4.2.1 Training model Word2Vec-ESN classifier

To train the Word2Vec-ESN classifier, training sentences are presented to the model sequentially. Figure 4.4 shows the processing of an example sentence by Word2Vec-ESN classifier. An input sentence is processed as many time as there are verbs in a sentence, forming multiple sequences. Thus the classifier takes an argument-predicate pair across the time as an input. Unlike Word2Vec-θRARes model, the readout layer has five neurons each coding for a role (P, A, O, R, XX). Thus during the training, the role of the input argument-predicate pair is also teacher-forced to the model. An output neurons have an activation 1 if an input argument-predicate pair has the corresponding role, -1 otherwise. The Word2vec model initially receives an argument-predicate pair of the input sentence and generates the distributed embeddings for both the input words. The generated word embeddings are concatenated and then taken by ESN as an input. Thus the size of ESN input layer is $2 \times E_v$ where the first E_v neurons takes the vector representation of the argument and remaining E_v neurons for the predicate. The reservoir internal states are collected for an input sequence over time. Reservoir-to-readout (W^{out}) weights are then learned by using the linear or ridge regression on collected reservoir states and the readout activations.

4.2.2 Decoding Output

Recall that the Word2Vec-ESN classifier process a sentence as many times as there are verbs in the sentence. Thus while testing, the output activations of the model is used to predict the role for an argument-predicate pair. The role having the highest output activation is considered as the role of an argument-predicate pair [3]. The role for all argument-predicate pairs is collected and the meaning of the sentence with respect to a verb can then be interpreted by filling up the tagged words in form "P(A, O, R)". For example, in the sentence "John hit the ball." as shown in figure 4.4, the roles for each word-verb pair (i.e. P, A, O, XX) is used to deduce the meaning of the sentence as "hit(John, ball, –)".

4.2.3 Evaluation Metrics

To analyze the performance of this Word2Vec-ESN classifier on TRA task, the confusion matrix or contingency table [1] is used and the classification scores: Accuracy, Precision, Recall, and F1-Score, are calculated for all possible roles. Higher the classification scores the better. To get the classification score of the model, scores of individual roles are macro-averaged, to get single real numbered scores. The reason for choosing macro-average is that it gives equal weights to all the roles, addressing the role imbalance problem [37]. The same evaluation metrics was also used for CoNLL-04 and CoNLL-05 semantic role labeling shared task [10, 9].

Table 4.1: Confusion matrix to evaluate Word2Vec-ESN classifier on TRA task.

		Predicted Roles				
		A	**O**	**R**	**P**	**XX**
True Roles	**A**	TP_A	E_{A-O}	E_{A-R}	E_{A-P}	E_{A-XX}
	O	E_{O-A}	TP_O	E_{O-R}	E_{O-P}	E_{O-XX}
	R	E_{R-A}	E_{R-0}	TP_R	E_{R-P}	E_{R-XX}
	P	E_{P-A}	E_{P-0}	E_{P-R}	TP_P	E_{P-XX}
	XX	E_{XX-A}	E_{XX-O}	E_{XX-R}	E_{XX-P}	TP_{XX}

The confusion matrix describes the predictions made by the model. The rows of the matrix correspond to the actual roles and the columns correspond the predictions made by the model. The top left to bottom right diagonal elements of this matrix represent the number of words for which the predicted role is equal to the actual role. So, higher the value of diagonal elements the better. The non-diagonal elements of the matrix represent the number of words which are incorrectly labeled.

Using the confusion matrix, accuracy can be calculated as the ratio of the number of correctly labeled words to the total number of words (equation 4.2.1). This accuracy measure specifies how often the classifier is correct [47].

$$Accuracy = \frac{\text{number of words correctly labelled}}{\text{total number of words}} \tag{4.2.1}$$

However, accuracy measure can be distorting (because of accuracy paradox), when the dataset has words with significant role imbalance as it gives high scores to models which merely predict the most frequent class. Thus accuracy measure cannot be used alone to evaluate the performance of the model [49, 53]. So, additional performance measures such as Precision (P), Recall(R), and F1-score (F1) are required to evaluate the model. All these measures take a value between 0 and 1.

Precision is defined as the ratio of True positive (TP) to False Positive(FP) and True Positive (equation 4.2.4). It is the measure of the accuracy of a role provided that a specific role has been predicted [47].

$$Precision = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{4.2.2}$$

From the confusion matrix above, the precision for the role Agent(A) is be calculated as:

$$Precision(A) = \frac{TP_A}{(TP_A + E_{O-A} + E_{R-A} + E_{P-A} + E_{XX-A})}$$

Recall is defined as the ratio of True Positive to True Positive and False Negative. It measures how good the model is in labeling the correct roles. It is also called *'Sensitivity'* or *'True Positive Rate'*. [47].

$$Recall = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \tag{4.2.3}$$

Recall for the role 'A', from the above confusion matrix is calculate as:

$$Recall(A) = \frac{TP_A}{(TP_A + E_{A-O} + E_{A-R} + E_{A-P} + E_{A-XX})}$$

F1-Score or (F1) is the harmonic mean of precision and recall. In other words, it represents the balance between the precision and recall. The F1-score measure takes the false positive and the false negative into consideration. This score is really useful whenever there is a class imbalance in the dataset [47] and is calculated as:

$$F1 = 2 \times \frac{\text{Precision} \times Recall}{\text{Precision + Recall}} \tag{4.2.4}$$

Chapter 5

Experiments

This chapter presents the experimental setup and experiments performed in this research work. The rest of the chapter is organized as follows. First, we will have an overview of the corpora used to carry out the experiments. Then we describe the configuration of the model necessary to perform the TRA task with the Word2Vec-θRARes model and the Word2Vec-ESN classifier proposed in the previous chapter. We will also see the experimental setup to perform the TRA task.

5.1 Corpora and pre-processing

This section describes the corpora used to perform experiments with the proposed Word2Vec-θRARes language model and Word2Vec-ESN classifier for TRA task. Firstly, we will have an overview of the corpora used for TRA task and then we will see the corpus used to train Word2Vec model.

5.1.1 Corpora For TRA Task

In order to have a fair comparison between the Word2Vec-θRARes model and θRARes model on the TRA task, the same corpora [1] used by Hinaut et al. [20, 21] with θRARes model was utilized. Thus, the corpus-373, corpus-462, corpus-90582 containing 373, 462 and 90582 sentences respectively, were used to perform the experiments with the proposed model on TRA task.

Corpus-45: Corpus-45 is a small corpus of 45 sentences. This corpus contains the constructions in grammatical forms i.e. 'N' and 'V' tokens were used to represent the nouns and verbs, along with the coded meaning of each sentence. The corpus contains the constructions with different grammatical structures i.e. active, passive, object-relative, subject-relative sentences which are represented in the form:

1. the N V the N . # N1-A1 N2-O1

[1] https://sites.google.com/site/xavierhinaut/downloads

2. the N was V by the N . # N1-O1 N2-A1

The coded meaning of the sentences is represented after the '#' token. The coded meaning 'N1-A1' can be interpreted as; the first noun is the agent of the first verb. For the object and the recipient roles 'O' and 'R' tokens were used respectively.

Corpus-462 and Corpus-90582: The sentences in the corpus-462 and corpus-90582 were generated by Hinaut et al. using a context-free grammar for English language and used for TRA task [20]. Each sentence in these corpora have verbs which take one, two, or three clause elements. For example, the sentences, "The man *jumps*", "The boy *cuts* an Apple", "John *gave* the ball to Marie", have verbs with the 1 (agent), 2 (agent and object) and 3 (agent, object and recipient) clause elements respectively. The sentences in the corpora have a maximum of four nouns and two verbs [20]. A maximum of one relative clause is present in the sentences; the verb in the relative clauses could take 1 or 2 clause elements (i.e., without recipient). For example, "The dog that bit the cat chased the boy". Both the corpus-462 and corpus-90582 have the constructions in the form:

1. walk giraffe $< o >$ AP $< /o >$; the giraffe walk -s . # ['the', 'X', 'X', '-s', '.']
2. cut beaver fish , kiss fish girl $< o >$ APO , APO $< /o >$; the beaver cut -s the fish that kiss -s the girl . # ['the', 'X', 'X', '-s', 'the', 'X', 'that', 'X', '-s', 'the', 'X', '.']

Each construction in the corpus was divided into four parts. The first part describes the meaning of the sentence using the semantic words (or open class words) in order of predicate, agent, object and recipient. The second part (between '$< o >$' and '$< /o >$') describes the order of the thematic roles of semantic words as they appear in the raw sentence. The third part (between ';' and '#') contains the raw sentence with verb inflections (i.e. '-s') and the fourth part is the abstract representation of a sentence with semantic words removed and replace with token 'X' [20].

Corpus-90582 have 90582 sentences along with the coded meaning of each sentence. This corpus is redundant; multiple sentences with different grammatical structure but the same coded meaning (see fig. 3.3). In total, there were only 2043 distinct coded meanings [20]. This corpus also has an additional property that along with complete coded meanings of the sentences, it also have incomplete meanings. For example, the sentence "The Ball was given to the Man have no *'Agent'*, and thus the meaning of the sentence is "given(-, ball, man)". The corpus also contains 5268 pair and 184 triplets of ambiguous sentences, i.e. 10536 and 553 sentences respectively. Thus in total there were 12.24% (i.e., $5268 \times 2 + 184 \times 3 = 11088$) of ambiguous sentences which have the similar grammatical structure but different coded meaning [20].

Corpus-373: Apart from the corpus-462 and corpus-90582 which were artificially constructed using the context-free grammar, the corpus-373 containing real sentences collected from humans in natural language was also used. Corpus-373 includes the instructions collected from the participants interacting with a humanoid robot (iCub) in a Human-Robot Interaction (HRI) study of language acquisition conducted by Hinaut et al. [21]. In the study, the robot first performs one or two actions by placing the available objects to a location (e.g. left, right, middle) and the participants observes the actions. Then the participants were asked to instruct the robot again to perform the same actions in the natural language. Thus the corpus contains 373 complex instructions to perform single or double actions with temporal correlation (see Action Performing task in Hinaut et al. [**?**] for more details). For example, the instruction "Point to the guitar" is a one action command whereas the instruction "Before you put the guitar on the left put the trumpet on the right" is a complex command with double actions, where the second action is specified before the first action. A list of 86 closed class words used to filter semantic words from the sentences is also provided with this corpus. Also, unlike corpus-462 and corpus-90582 this corpus does not contain verbs inflections. Thus, this data is complex enough to test the learnability and generalization of the proposed models on TRA task.

Pre-processing: As described earlier, unlike the θRARes model, the proposed models process the raw sentences. Thus the sentences in corpus-45 are manually pre-processed by replacing the token 'N' and 'V' with appropriate nouns and verbs such that the coded meaning of the sentence is not changed. For example, the construction "the N V the N" was changed to "the man pushed the ball". Recall that the corpus-462 and corpus-90582 contains the sentences where the verbs are represented along with inflections (suffixes "-s", "-ed", "-ing"). So these constructions in corpus-462 and corpus-90582 were processed to obtain the raw sentences without verb inflections. Firstly, all the words are lowercased and then the verbs with inflections are replaced by conjugated verbs [2]. The verb conjugation to be used depends on the inflection used for the verb. For example, the sentences "The giraffe walk -s" and "John eat -ed the apple" has been changed to "The giraffe walks" and "John ate the apple" respectively. This preprocessing was done because the distributed word representation captured by Word2Vec model already captures these syntactic relations which was imposed previously using verb inflections e.g $vector('walks') - vector('walk') \approx vector('talks') - vector('talk')$.

5.1.2 Corpus For Training Word2Vec Model

In this work, to train the Word2Vec model, a general purpose dataset i.e. Wikipedia corpus [3] ($\approx$ 14 GB) was used to obtain the low dimensional distributed embeddings of words. The corpus contains 2,65,8629 unique words. The reason for choosing the

[2]service used to find verb conjugations: http://www.scientificpsychic.com/cgi-bin/verbconj2.pl
[3]https://dumps.wikimedia.org/enwiki/latest/

Wikipedia dataset to train the model is that it contains most of the words used in the English language. Also, the Word2Vec model does not give good quality vector representation for words when trained over a small corpus thus a general purpose data set with billions of words is required to have better word embeddings. Thus more words we have the better the vector representation of words. Once the vector representation of words in the Wikipedia dataset is obtained, the model can then be trained further on our domain specific dataset (corpus-462 and corpus-90582) with more bias toward domain specific dataset to update the previously learned vector representations.

5.2 Models Configuration for TRA task

In this section, we will see the necessary configuration required for the proposed model. Both the Word2Vec-θRARes and Word2Vec-ESN classifier are initialized in the same way as described in the following subsections.

5.2.1 Obtaining Word Embeddings

Before using the proposed models for TRA task, the distributed embeddings of the words are required to be learned by Word2Vec model. To get the vector representation of words, the Word2Vec model is first trained on the Wikipedia dataset using skip-gram negative sampling approach (see Chapter 2 for more details). Skip-gram negative sampling approach was used because it was claimed to accelerate the training of the model and the resultant word vectors performs better on word analogy task[34, 33].

To obtain the word vectors of different dimensions, six different Word2Vec models were trained. The hidden layers of each model contain 20, 30, 50, 100, 200, 300 neurons respectively. Thus each model produces the distributed word vectors corresponding to the size of the hidden layer. All the six models were trained using the same hypermeters as follows. A context window of ± 5 for the current input word was used. The negative sampling size of 5 was chosen i.e. 5 noise words are chosen randomly from the vocabulary which does not appear in the context of the current input word. Additionally, the most frequent words are discarded with the probability using equation 2.1.10 with a subsampling threshold of $t = 10^{-5}$. All the words which appear less than 5 times in the corpus were also ignored. To update the network weights stochastic gradient descent was used [44, 33] with initial learning rate of $\alpha = 0.025$, which drops to $min_alpha = 0.0001$ linearly as training progress.

The word embeddings obtained by training the Word2Vec model on Wikipedia dataset are accurate enough to capture the semantic relationship of the words for e.g. $vec(`Paris') - vec(`France') + vec(`Germany') \approx vec(`Berlin')$. Before training the model on Wikipedia corpus, a vocabulary of words in the corpus is created. Once the vocabulary is created, and the model is trained, it was not possible to add new words to this vocabulary. However, there is a possibility

that when a domain specific corpus (e.g. corpus-462, corpus-373, etc.) is used to further train the Word2Vec model, some of the words may not be present in previously generated vocabulary. Due to this limitation, it was not possible to get the distributed embeddings of these new words. Thus we needed to update the vocabulary of the model if some words are not already present in the vocabulary to facilitate the online training of Word2Vec model. Unfortunately, neither the C++ API [4] nor the Gensim python API [43] implementation of Word2Vec model supports the vocabulary update, once created. So, we implemented the online training [5] of Word2Vec by modifying and extending the Gensim API. The new words that were not present in the existing vocabulary can now be added and initialized with some random weights, and the model can then be trained in the usual manner to learn the distributed embeddings of new words. Although now the vocabulary can be updated in an online manner, the vector embeddings of the newly added words have poor quality if their count in the new corpus is less. Thus, to have more impact of the domain specific corpus on the words embeddings and to improve the quality of words vectors, the domain specific dataset was repeated several times before training the model [6].

So now when we have an online version of training the Word2Vec model, the training of Word2Vec model can be resumed on the domain-specific corpora (corpus-462 and corpus-90582). While updating the model on the new dataset, no word was disregarded irrespective of its count in the corpora. This allows us to have the vector embeddings of all the words in our domain specific corpora. Once the Word2Vec model is trained, the generated word embeddings were normalized using L-2 norm before using them for TRA task. One important point to remember while training word2vec model is that if an already trained model has to be trained further on any other corpus, then normalization should not be done, as it is not possible to train the model again with normalized word vectors [43]. The trained Word2Vec model is now ready to be used with the Word2Vec-θRARes model and Word2Vec-ESN classifier.

5.2.2 ESN reservoir initialization

The size of the reservoir is one of the critical parameters of ESN and it is often recommended to use a bigger reservoir that can be computationally afforded, provided the appropriate regularization method is used [32]. Thus for the proposed models, a reservoir of 1000 (until and unless specified) leaky integrator neurons with *tanh* activation function was used. The input-to-hidden (W^{in}), hidden-to-hidden (W^{res}) weights were randomly and sparsely generated using a normal distribution with mean 0 and variance 1. For the TRA task, output-to-reservoir weights (W^{back}) was not used. The reservoir state update equations 2.2.2 and 2.2.3 thus changes to:

[4] https://code.google.com/archive/p/word2vec/

[5] The code is adapted from- http://rutumulkar.com/blog/2015/word2vec

[6] Idea discussed on: https://groups.google.com/forum/#!topic/gensim/Z9fr0B88X0w

$$x'(n) = tanh\ (W^{res}x(n-1) + W^{in}.u(n)) \tag{5.2.1}$$

$$x(n) = (1-\alpha)x(n-1) + \alpha x'(n) \tag{5.2.2}$$

To generate the sparse weights a fixed fanout number of $F_{hh} = 10$ and $F_{ih} = 2$ was used i.e. each reservoir neuron was randomly connected to 10 other reservoir neurons and each input neuron was connected to only 2 other reservoir neurons respectively. Use of the fixed fanout number scales the computational cost of reservoir state update linearly with increase in reservoir size [32]. After the weights initialization, there are several other reservoir parameters which are to be optimized and are task specific. In the next subsection, we will see how these parameters are optimized.

5.2.3 Learning ESN parameters

Echo state network have several parameters to be optimized for the proposed models to perform efficiently on the TRA task. Some of the parameters like reservoir size, sparsity, distribution of non-zero weights are straightforward [32]. Whereas other parameters like Spectral Radius (SR), Input Scaling (IS) and Leak Rate (LR) are task dependent and are often tuned by multiple trials. Thus to identify these parameters for TRA task, a grid search over the parameter space using five reservoir instances was performed. As the parameter search space can be gigantic, a broader grid with wider parameter ranges was first explored to find the optimal grid: grid with low sentence error for Word2Vec-θRARes model and high F1-Score for the Word2Vec-ESN classifier. The optimal region identified during the broader grid search was then used for the narrow search to identify the optimal parameters [32]. As both the proposed models process the sentences differently and have different training objectives, the ESN parameters for both the models were optimized separately. Additionally, the Word2Vec-θRARes model parameters are optimized separately for SCL and SFL mode. In this section, we will see the parameter optimization on corpus-462, as most of the experiments were performed using this corpus.

Word2Vec-θRARes model: To optimize the reservoir parameters, corpus-462 with the topologically modified coded meaning was used. The choice of using the modified coded meaning was deliberate and will be made clear later in Experiment-6. To get the optimal parameters (i.e. parameters resulting in the lowest sentence error) of Word2Vec-θRARes model, the model was trained and tested over a range of parameters using 10-fold cross-validation method. The corpus-462 with 462 sentences was randomly split into ten equally sized subsets (i.e. each subset with $\approx$ 46 sentences). The model was trained on sentences from 9 subsets and then tested on remaining one subset. This process was repeated ten times such that the model was trained and tested on all the subsets at least once. A reservoir of 1000 neurons and a fixed regularization coefficient, $\beta = 1e{-}6$ for ridge regression was used. By exploring the parameter space we identified the optimal parameters

as $SR = 2.4$, $IS = 2.5$ and $LR = 0.07$ in SCL mode and $SR = 2.2$, $IS = 2.3$ and $LR = 0.13$ in SFL mode.

Word2Vec-ESN classifier: For the Word2Vec-ESN classifier, the optimal parameters (i.e. parameters resulting to highest F1-Score) were identified during the grid search using the same corpus-462 and by applying 10-fold cross-validation. Keeping all the other conditions i.e. reservoir size and regularization coefficient, identical to Word2Vec-θRARes model described above, optimal parameters were identified as $SR = 0.7$, $IS = 1.15$, $LR = 0.1$. As later in the Experiment-2, the performance of the Word2Vec-ESN classifier will also be tested on the sentences transformed to grammatical form (i.e. semantic words replaced with 'SW' token) and represented in localist fashion. Thus it is also required to find the optimal parameters for this transformed corpus. Keeping all the conditions same, the grid search was performed over parameter space and identified the optimal parameters were identified as $SR = 1.3$, $IS = 1.0$, $LR = 0.4$.

5.2.4 Input and Output Coding

As specified in chapter 4, the Word2Vec-θRARes model and Word2Vec-ESN classifier process the sentences differently, thus the input and output coding for both the model also differs. However, the initialization of Word2Vec model and ESN reservoir weights remains same for both the models.

Word2Vec-θRARes model: A raw sentence is presented to the model, where each word in the sentence is processed across time by both Word2Vec model and ESN. The Word2Vec model outputs the $E_v = 50$ dimension word embedding which is then used as an input for ESN. Thus input layer of ESN has 50 neurons.

For all the experiments on corpus 45, 462 and 90582 (Experiments 1-5 in next section), an equivalent but topologically modified coded meaning (see section 4.2) was used. The readout layer of ESN contains 24 ($4 \times 3 \times 2$) neurons as the corpus contains sentences, having a maximum of 4 nouns each having 3 possible roles (Agent, Object, and Recipient) with respect to a maximum of 2 verbs. Each neuron in the readout layer thus codes for a role of a noun with respect to a verb. The corpus-90582 contains a maximum of 5 nouns, and thus the size of readout layer is 30 ($5 \times 3 \times 2$).

For Experiment-6, the topologically modified coding was not used. So, the readout layer contains 36 ($6 \times 3 \times 2$) neurons as there are maximum of 6 semantic words in this corpus and each could take one of the 3 possible thematic roles (Object, Predicate, Location) with respect to a maximum of two actions [21]. Output neurons have an activation 1 if the corresponding roles are present in the sentence, -1 otherwise.

Word2Vec-ESN classifier: Recall that in Word2Vec-ESN classifier a raw sentence is presented to the model, where each word (argument) along with the verb

(Predicate) with respect to which the word is currently processed, is input to the model across time (see section 4.2). Also, A sentence is processed as many time as there are verbs in the sentence. So, the Word2Vec model takes this argument-predicate pair as an input and outputs a vector of $E_v = 2 \times 50$ dimension, which is then used as an input to ESN. Thus the input layer contains 100 neurons where first 50 neurons encodes the vector representation of the word and remaining 50 neurons codes for the verb with respect to which word is being processed. Unlike the Word2Vec-θRARes, the size of readout layer always remains the same and contains five neurons each coding for a role: Predicate (P), Agent (A), Object(O), Recipient (R) and No Role (XX) for both corpus-462 and corpus-90582. Readout neuron of ESN has an activation 1, if the input word-verb (argument-predicate) pair have the corresponding role, -1 otherwise.

5.3 Experimental Setup

In this section we descibe the experiments performed in this thesis. Untill and unless specified we use all the optimal parameters identified during the grid search for both Word2Vec-θRARes model and its Word2Vec-ESN classifier, described in section 5.2.3.

5.3.1 Experiment-1: Models performance on small corpous

In order to see the performance of Word2Vec-θRARes model and its Word2Vec-ESN classifier on TRA task with the small corpus, the model was first tested with the limited number of sentences i.e. 26 sentences (sentence 15 to 40) from corpus-45. The chosen sentences have distinct surface form and grammatical structure (e.g. active, passive, dative-passive). Both the models were first trained and tested on all the sentences to get the training errors and then tested using Leave one Out (LoO) cross-validation method: training on 25 sentences and testing on remaining one sentence such that all the sentences are tested at least once. Reservoir parameters for this experiments were optimized on the 26 constructions used in the experiment by exploring the parameter space and using LoO cross-validation approach. This experiment remains a toy demonstration, and we will later see the generalization capability of the model with an extended corpus in Experiment-2 and Experiment-5.

Word2Vec-θRARes model: For simulation with Word2Vec-θRARes model, five reservoir instances each with 1000 neurons and reservoir parameters $SR = 2.1$, $IS = 6.6$, $LR = 0.5$ was used. The model was operated in SCL mode.

Word2Vec-ESN classifier: For simulations with the Word2Vec-ESN classifier, five reservoir instances each with 500 neurons and reservoir parameters $SR = 0.2$, $IS = 0.7$, $LR = 0.7$, was used.

5.3.2 Experiment-2: Generalization Capabilities

In the experiment 5.3.1, we performed simulations with the limited set of sentences. Thus in order to test the generalization capability of the model, an extended corpus of 462 sentences was examined (see corpus-462 in **??**).

Word2Vec-θRARes model: Using the extended corpus-462, generalization capability of Word2Vec-θRARes model was tested using 10-fold cross-validation approach. For this experiment, equivalent but topologically modified output coding of sentences was used for training (see fig. 4.4). In order to compare the generalization ability of the Word2Vec-θRARes model and θRARes model, simulations were performed in both the learning modes i.e. in SCL and SFL mode. The reservoir parameters identified during grid search for both the learning modes were used i.e. $SR = 2.4$, $IS = 2.5$, and $LR = 0.07$ in SCL mode and $SR = 2.2$, $IS = 2.3$ and $LR = 0.13$ SFL mode (see section 5.2.3). For simulations in each learning mode, five reservoir instances (i.e. reservoir weights were initialized using 5 different random generator seeds) each with a reservoir of size 1000 neurons was used.

Word2Vec-ESN classifier: Recall that the Word2Vec-ESN classifier unlike Word2Vec-θRARes model evaluate the performance of model using classification metrics(see section 4.2 in chapter 4). The model process the raw sentences and the distributed vector representation of input argument-predicate pair is fed into ESN. However, it is also important to test the behavior of this model on the sentences transformed to grammatical form and using localist word representation i.e. without the word2vec unit in the model, as an input to ESN. This will allow us to have an insight on the benefits of distributed word vector. Thus for this experiment, the Word2Vec-ESN classifier was used in two configurations mentioned below. Simulations in both the configurations were performed using five reservoir instances each with a reservoir of 1000 neurons.

1. **Configuration-1:** In this configuration, the Word2Vec-ESN classifier process the raw sentences and distributed vector representation of argument-predicate pairs are input to ESN (see section 4.2 for more detail). For the simulation in this configuration, the reservoir parameters $SR = 0.7$, $IS = 1.15$, $LR = 0.1$ were used. The parameters are identified previously by exploring the parameter space.

2. **Configuration-2:** In this configuration, the Word2Vec-ESN classifier is used without Word2Vec unit, and only ESN is used for processing the sentences. The input sentences were transformed in the grammatical form: all the semantic words in the sentence were replaced with 'SW' token, and localist word vectors are input to ESN. For the simulation in this configuration, the reservoir parameters $SR = 1.3$, $IS = 1.0$, $LR = 0.4$ found previously during grid search were used.

5.3.3 Experiment-3: Effect of Corpus structure

As described earlier, that the sentences in the corpus-462 were created based on context-free grammar. Thus the sentences in the corpus contain an inherent grammatical structure. The models thus possibly utilize the underlying grammatical structure to some extent for learning and generalizing. To test this hypothesis and to demonstrate that the model is not generalizing on any other inconsistent regularity in the corpus, in this experiment, the inherent grammatical structure from the sentences was removed by randomizing the word orders within the sentences [20]. Such a test will also help us to have an insight on what the model is actually learning and whether the model is overfitting or not.

Word2Vec-θRARes model: In this experiment, the corpus-462 with the scrambled sentences (i.e. in the absence of any grammatical structure) was presented to Word2Vec-θRARes model. Keeping all the conditions and the reservoir parameters same as used in Experiment-2, a 10-fold cross-validation was performed with five model instances. The cross-validation errors obtained in the Experiment-2 on the corpus-462 with inherent grammatical structure can then be compared with the cross-validation error obtained while using scrambled corpus. If the model is not overfitting and learning from the grammatical structure, then the model should not perform better on the corpus with scrambled sentences (i.e. in the absence of grammatical structure). However, in the case of overfitting the generalization effect on the corpus should not vary much both in presence and absence of grammatical structure [20].

Word2Vec-θRARes model: Like Word2Vec-θRARes model, the Word2Vec-ESN classifier was also presented with the scrambled sentences of corpus-462. Keeping all the conditions and the reservoir parameters same as used in Experiment-2, a 10-fold cross-validation was performed with five instances of the Word2Vec-ESN classifier. The Word2Vec-ESN classifier was also operated with scrambled corpus in configuration-1 and configuration-2 as described in Experiment-2.

5.3.4 Experiment-4: Effect of Reservoir size

The most important hyperparameter which affects the performance of the model is the size of the reservoir (i.e. N_x = number of neurons in the reservoir). The addition of neurons in the reservoir is also computationally inexpensive because the read-out weights (W^{out}) scales linearly with the number of neurons in the reservoir [51]. So, in order to determine the effect of reservoir size on the performance of the Word2Vec-θRARes model and Word2Vec-ESN classifier, the simulations were performed over a range of reservoir sizes [1] [2] using five reservoir instances.

[1] Reservoir sizes explored in Word2Vec-θRARes model: [90, 291, 493, 695, 896, 1098, 1776, 2882, 3988, 5094]

[2] Reservoir sizes explored in Word2Vec-ESN classifier: [50, 100, 250, 400, 600, 800, 1050, 1600, 2250, 3320, 3860, 4500]

Word2Vec-θRARes model: With Word2Vec-θRARes model, the simulations were performed on a range of reservoir size[1] in both SCL and SFL modes using the same optimal reservoir parameters as used in the Experiment-2 i.e. $SR = 2.4$, $IS = 2.5$ and $LR = 0.07$ in SCL mode and $SR = 2.2$, $IS = 2.3$ and $LR = 0.13$ in SFL mode.

Word2Vec-ESN classifier: To see the effect of reservoir size on Word2Vec-ESN classifier, simulations were performed on a range of reservoir sizes [2], in both the configuration-1 and configuration-2 of Word2Vec-ESN classifier as described in Experiment-2.

5.3.5 Experiment-5: Scalability of the model

In order to investigate the effect of corpus size and determine the scaling capability of the model, the extended corpus-90582 (see section 5.1) was used for this experiment. As this corpus also contains 12% of ambiguous sentences which impedes the learning and generalization of the model, this experiment will also validate the model's ability to process the abmbiguous sentences. Thus, in order to study the scaling capabilty of the model with different corpus size, 6 sub-corpora were created by randomly sampling 6%, 12%, 25%, 50%, 75%, 100% of sentences from the orginal corpus of 90582 sentences [20].

Word2Vec-θRARes model: Each of the generated sub-corpora was exposed to the Word2Vec-θRARes model, and a 2-fold cross-validation was performed. The model was trained on half the sub-corpora size and tested on remaining half. The second half used for testing was then used to train the model and tested on the first half, used for training previously. This experiment was performed with the Word2Vec-θRARes model in both the learning modes i.e. SCL and SFL mode, using five reservoir instances each with 5000 neurons. All the other parameters were kept identical to the Experiment-2.

5.3.6 Experiment-6: Generalization on new corpus

One may argue that the previously used corpora (corpus-462 and corpus-90582), which were artificially constructed using context-free grammar may add a bias to the model. Thus making it easier for the model to learn and generalize on these corpora for TRA task. To answer this question, in this experiment, the corpus-373 (see section 5.1) containing the instructions collected from humans in a Human-Robot Interaction (HRI) study of language acquisition was used.

Word2Vec-θRARes model: To test the generalization of Word2Vec-θRARes model on corpus-373, the model was operated in SCL mode and tested using the LoO cross-validation method. The SCL mode and the LoO method were chosen so that results of this experiment can be compared with that of obtained using

θRARes model [21]. Also, to make the results of both the model deterministically comparable, topologically modified coding was not utilized for this experiment (see section 4.1) and simulations were performed using ten reservoir instances.

For this experiment, reservoir parameter space was not explored to identify optimal reservoir parameters, but instead, the previously optimized parameters obtained on corpus-462 with topological modified coded meaning was used i.e. $SR = 2.4$, $IS = 2.5$ and $LR = 0.07$ (see section 5.2.3). Doing so will enable us to test the robustness of these model parameters on the new corpus i.e. corpus-373.

5.3.7 Experiment-7: Effect of Word2Vec word dimensions

In all the previous experiments, the distributed word vectors of 50 dimension were used. Mikolov et al. [34, 33] showed that the word vectors of higher dimensions (e.g. 300 in their case) perform better on word analogy task. Thus in this experiment, the effect distributed word vector dimensions on the TRA task will be explored. This experiment was only performed on Word2Vec-θRARes language model.

Word2Vec-θRARes model: Recall that in section 5.2.1, six Word2Vec models were trained to generate the word vectors of 20, 30, 50, 100, 200, 300 dimensions respectively. To test the effect of these word vector dimensions on TRA task, each of these Word2Vec model was used in Word2Vec-θRARes language model. For this experiment, the corpus-373 without the topologically modified coded meaning was used (see fig. 4.2). Simulations were performed using ten reservoir instances each with 1000 neurons. The model was trained and tested using LoO cross-validation approach in both the SCL and SFL mode, and all the reservoir parameters were kept identical to Experiment-2.

Chapter 6

Results and Discussion

This chapter summarizes the results of the experiments performed in this research work and described in the previous chapter. The results are presented in the same order of the experiments performed. In this chapter, the results obtained are also discussed and compared with that of obtained using θRARes model.

6.1 Experiment-1: Models performance on the small corpus

Word2Vec-ESN Model: When trained and tested on all the 26 sentences, the model learned the sentences without any errors. But during the cross-validation, the model generalized with 35.09% meaning error and 85.38% sentence error. As illustrated in Table 6.1, it can be seen that with Word2Vec-θRARes model failed to generalize well on the limited set of 26 constructions. Comparing the performance of the Word2Vec-θRARes model with the θRARes model, the generalization errors with Word2Vec-θRARes model are almost doubled on the same set of 26 constructions.

It is also important to know that the results of θRARes model reported in the table 6.1 are averaged over 100 reservoir instance of 100 units each [20]. Whereas results of Word2Vec-θRARes are averaged over five reservoir instances of 1000 neurons each. With the small reservoir of 100 neurons, the Word2Vec-θRARes failed to learn and generalize on the 26 sentences (not reported here).

Table 6.1: Generalization errors in SCL mode on 26 sentences of corpus-45.

		Word2Vec-θRARes	θRARes
Meaning Error	mean	35.09	14.83
	std.	05.89	02.59
Sentence Error	mean	85.38	42.73
	std.	07.88	07.19

LoO cross-calidation errors with both Word2Vec-θRARes model and θRARes model (shown in %). Mean and std. is average and standard deviation of errors over several model instances.

Word2Vec-ESN classifier: As illustrated in Table 6.2, the Word2Vec-ESN classifier learned the 26 sentences resulting in F1-Score of 95.69%. While testing using LoO cross-validation method the model resulted into low F1-Score of 68.53%. The high difference in F1-Score during training and testing points toward the limited generalization on unseen sentences.

Table 6.2: Classification score of model Word2Vec-ESN classifier on 26 sentences from corpus-45.

		Precision	Recall	F1-Score
train	mean	94.92	96.54	95.68
	std.	00.27	00.74	00.46
test	mean	69.87	71.29	68.53
	std.	02.36	01.37	01.80

LoO classification scores of Word2Vec-ESN classifier(shown in %). Mean and std. is average and standard deviation of classification scores over several model instances.

6.2 Experiment-2: Generalization Capabilities

Word2Vec-θRARes model: When the model was initially trained and tested on all the 462 sentences of corpus-462, it learned the sentences with 0.55% meaning error and 1.64% sentence error in the SCL mode and 0.16% meaning error and 0.52% sentence error in the SFL mode. Using the 10-fold cross-validation, the model generalized to 8.68%($\pm$1.01%) meaning error and 24.09%($\pm$2.38%) sentence error in the SCL mode. Whereas in the SFL mode the optimal meaning and sentence error were observed as 8.88%($\pm$0.14%) and 25.17%($\pm$0.01%) respectively.

As illustrated in Table 6.3, one can see that while training both the Word2Vec-θRARes and θRARes model learned all the sentences thoroughly (less than 1% error rates) in SCL and SFL mode. Comparing the performance of both the model during testing, we can see an improvement of 8.04% sentence error in SCL mode with Word2Vec-θRARes model, whereas the meaning error dropped by 1.25%. However, considering the standard deviation, we can also say that the meaning error remained almost equivalent in both the Word2Vec-θRARes and θRARes model. In SFL mode, both the meaning and sentence errors remained nearly equal for both the models. It can also be observed that with Word2Vec-θRARes model, the performance of the model is improved mainly in SCL mode as compared to SFL mode, whereas it is vice-versa in θRARes model.

Table 6.3: Mean and standard deviation of meaning and sentence error on train and test set of coprus-462 in different learning modes.

		Word2Vec-θRARes				θRARes			
		Corpus 462		462 scrambled		Corpus-462		462 scrambled	
		ME	SE	ME	SE	ME	SE	ME	SE
SCL train	mean	0.55	1.64	6.68	26.80	0.12	1.21	4.81	20.43
	std	0.06	0.12	0.67	1.64	0.03	0.30	0.30	1.25
SCL test	mean	8.68	24.09	70.15	99.26	7.43	32.13	74.15	99.89
	std	1.01	2.38	1.24	0.29	0.52	1.35	0.80	0.15
SFL train	mean	0.16	0.52	9.38	35.58	0.00	0.00	0.00	0.00
	std	0.09	0.25	0.69	3.40	0.00	0.00	0.00	0.00
SFL test	mean	8.88	25.17	67.87	99.26	9.18	24.37	73.39	99.91
	std	0.14	0.01	0.10	0.33	0.57	1.19	0.96	0.11

Meaning (ME) and Sentence error (SE) in different learning modes with Word2Vec-θRARes model using distributed word embeddings and θRARes model [20] which uses grammatical form and localist representation of words of sentences. The errors are given in percentage with two decimal precision. SCL: Sentence Continuous Learning; SFL: Sentence Final Learning; std: standard deviations. Simulations were done with 5 model instances with the reservoir of 1000 neurons.

Word2Vec-ESN classifier: Table 6.4, illustrates the training and cross-validation classification scores of simulations performed with the Word2Vec-ESN classifier in both the configurations as described in Experiment-2 (see section 6.2). In configuration-1, the model when trained and tested on all the 462 sentences of corpus-462, the model learned to label the argument-predicate pairs in the sentences with the Precision (Pr), Recall (Re) and F1-Score (F1) of 97.46%, 92.29%, 94.65% respectively. When tested using 10-fold cross validation for generalization, the model genralized with Pr = 96.76%, Re = 91.78% and F1 = 93.99%. Notice

Table 6.4: Classification scores produced by Word2Vec-ESN classifier on unscrambled and scrambled corpus-462 in two configurations.

		Configuration-1		Configuration-2	
		coprus-462	462 scrambled	coprus-462	462 scrambled
Precision	test	96.76 (± 0.08)	95.72 (± 0.15)	61.91 (± 0.07)	16.71 (± 0.64)
	train	97.46 (± 0.00)	96.96 (± 0.13)	61.96 (± 0.07)	57.24 (± 7.90)
Recall	test	91.78 (± 0.08)	89.75 (± 0.11)	68.20 (± 0.46)	20.01 (± 0.02)
	train	92.29 (± 0.22)	90.40 (± 0.05)	67.69 (± 0.29)	20.17 (± 0.04)
F1-Score	test	93.99 (± 0.09)	92.28 (± 0.12)	63.71 (± 0.22)	17.49 (± 0.05)
	train	94.65 (± 0.00)	93.24 (± 0.08)	63.55 (± 0.17)	17.74 (± 0.08)

Classification scores (in %) of Word2Vec-ESN classifier during training and testing condition. Configuration-1: raw sentences are processed, and distributed vectors of the argument-predicate pair are input to ESN. Configuration-2: Sentences transformed to grammatical form and localist word vectors are input to ESN. Simulations were done with five instances of Word2Vec-ESN classifier each with a reservoir of 1000 neurons.

the marginal difference between the training and cross validation scores, indicating that the Word2Vec-ESN classifier in configuration-1 is generalizing well on untrained sentences and is not overfitting.

In configuration-2, where the sentences are transformed to GF and localist word representations as an input to ESN, the model learned the sentences with Pr = 61.96%(±0.07%), Re = 67.69%(±0.29%), F1 = 63.55%(±0.17%) during training. With 10-fold cross-validation the model generalized with Pr = 61.91%(±0.07%), Re = 68.20%(±0.46%) and F1 = 63.71%(±0.22%). It can be noticed that the Word2Vec-ESN classifier in configuration-2 did not perform well in comparision with configuration-1. Both the training and testing scores are much lower when compared to scores of configuration-1.

Figure 6.1 shows the confusion matrix, plotted using cross-validation results of the Word2Vec-ESN classifier in configuration-1 and configuration-2 (see section 6.2). The corresponding classification scores of individual roles produced by Word2Vec-ESN classifier during 10-fold cross-validation in both the configuration are also reported in Table6.5.

As seen in the confusion matrix (see fig. 6.1), when using the distributed vectors of argument-predicate pair as an input to ESN (i.e. configuration-1), the model predicted the words with the role 'Recipient', 'Predicate' and 'No Roles' almost without making any errors. The model predicted 78.6% of the words with the role 'Agent' correctly, whereas 3.9% and 17.5% of the words with the role 'Agent' were incorrectly predicted as 'Object' and 'No Role' respectively. Similarly, 81.4% of the words with the role 'Object' were correctly classified, whereas 5.1% of the words with actual roles as 'Object' were misclassified as 'Agent' and 13.5% as 'No Role'. On the other hand, only 0.2% the words with the actual role 'No Role' were

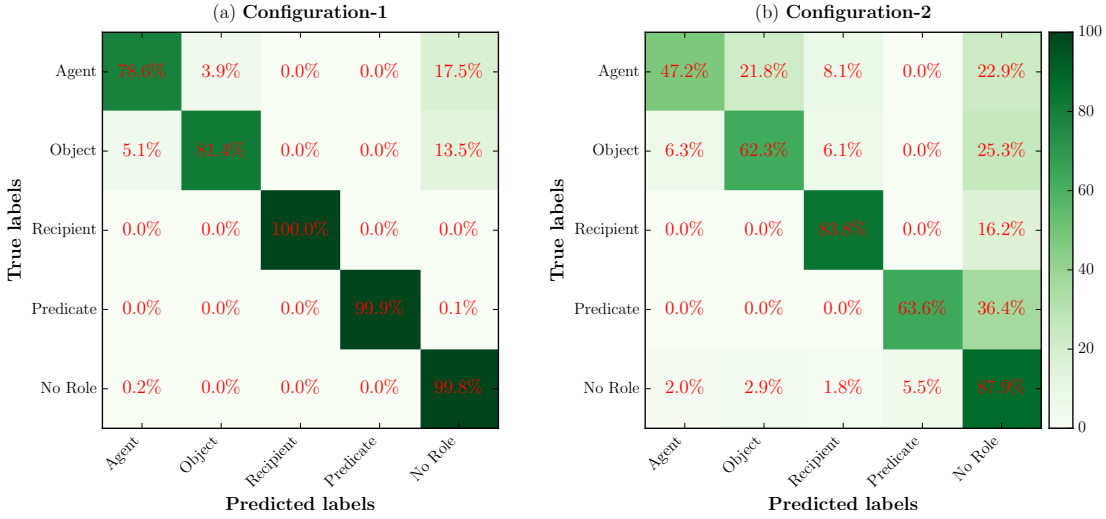

Figure 6.1: **Normalized confusion matrix with Word2Vec-ESN classifier in configuration-1 and configuration-2:**The confusion matrix with true roles (in rows) and predicted roles (in columns). The top-left to bottom-right diagonal shows the percentage of words whose roles are predicted correctly. Everything other than this diagonal represents the incorrect prediction of roles. Configuration-1: raw sentences processed by Word2Vec-ESN classifier and word2vec word vectors are input to ESN. Configuration-2: Sentences transformed to grammatical form and localist word vectors are input to ESN. The results were obtained with reservoir of 1000 neurons and 10 fold-cross validation.

wrongly predicted as 'Agent'.

Using the sentences in GF along with localist word vectors of the argument-predicate pair, as an input to ESN, the roles 'Recipient' and 'No Role' were correctly predicted with least errors of 16.2% and 12.1%. The Word2Vec-ESN classifier wrongly predicted all the roles as 'No Role', where the 'Predicate' being the highest role to be misclassified as 'No Role' (36.4%). The model seems to be confused with the words having roles 'Agent' and 'Object'. 21.8% of the words with the role 'Agent' were incorrectly predicted as 'Object', whereas 6.3% of words with the actual role as 'Object' were misclassified as 'Agent'. Also, the words with the roles 'Agent' and 'Object' were confused with the role 'Recipient'. 8.1% of the words with the role 'Agent' were wrongly predicted as 'Recipient' and 6.1% of the words with the role 'Object' were incorrectly predicted as 'Recipient'.

While comparing the predictions made by the Word2Vec-ESN classifier in both the configurations, it was observed that the roles 'Agent' and 'Object' made the most number of error in predictions as compared to other roles. In configuration-2, the roles 'Agent' and 'Object' were wrongly predicted as 'Recipient', whereas in configuration-1, this misprediction does not exist. Comparing the false predictions made by the Word2Vec-ESN classifier in both the configurations, it was observed that in configuration-1, the incorrect predictions were comparatively less. In configuration-1, all the words with roles 'Recipient', 'Predicate' and 'No Role' were

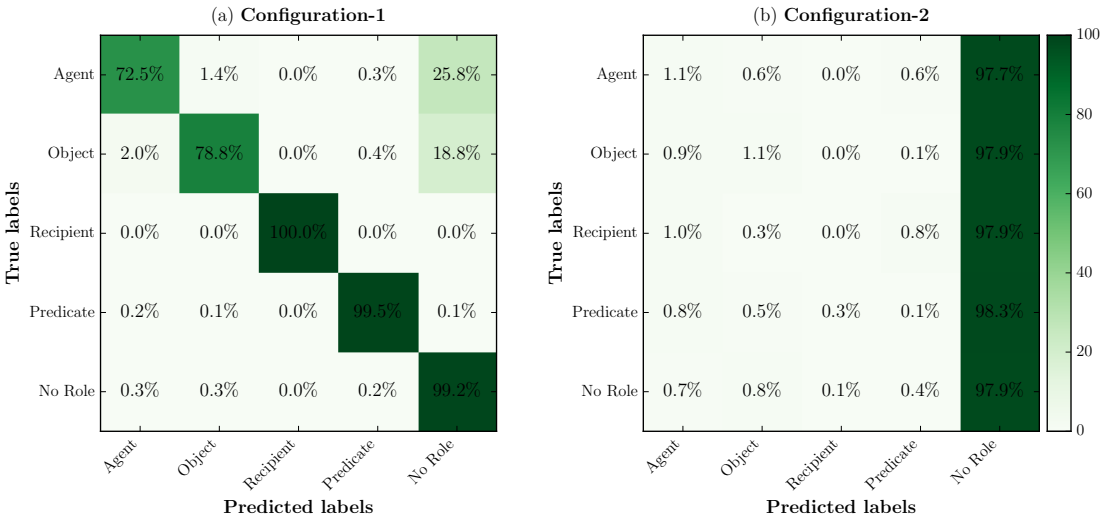

Figure 6.2: **Normalized confusion matrix with the Word2Vec-ESN classifier on scrambled corpus-462:** The confusion matrix with true roles (in rows) and predicted roles (in columns). The top-left to bottom-right diagonal shows the percentage of words whose roles are predicted correctly. Everything other than this diagonal represents the incorrect prediction of roles. Configuration-1: raw sentences processed by the Word2Vec-ESN classifier and word2vec word vectors are input to ESN. Configuration-2: Sentences transformed to grammatical form and localist word vectors are input to ESN. The results were obtained with reservoir of 1000 neurons and 10 fold-cross validation.

predicted almost without any errors whereas, in configuration-2, the Word2Vec-ESN classifier misclassified these roles respectively by 16.2%, 36.4%, and 12.1%.

6.3 Experiment-3: Effect of Corpus structure

Word2Vec-θRARes model: As illustrated in Table 6.3, it was observed that when using the scrambled corpus as an input to the Word2Vec-θRARes model, the model learned with low error rates of $ME = 6.68\%$ and $SE = 26.80\%$ in SCL mode. While testing, the model generalized with cross-validation error rates of $ME = 70.15\%$ and $SE = 99.26\%$. Similarly, in SFL mode the model learned with low error rates of $ME = 9.38\%$ and $SE = 35.58\%$ but during cross-validation high error rates of $ME = 67.87\%$ and $SE = 99.26\%$ was observed. Notice that the model learned the scrambled corpus with low error rates while training but during the cross-validation resulted into high error rates.

Word2Vec-ESN classifier: Table 6.4 illustrates the effect of scrambled corpus on the Word2Vec-ESN classifier. It can be seen that in the configuration-1, the classification scores remained almost equivalent to that obtained with the unscrambled corpus. Whereas in the configuration-2, the Word2Vec-ESN classifier failed

Table 6.5: Training and testing classification scores for individual roles when using Word2Vec-ESN classifier in two different configurations.

		Configuration-1			Configuration-2			
Role		Pr	Re	F1	Pr	Re	F1	Support
Agent	test	0.92	0.79	0.85	0.63	0.47	0.54	888
	train	0.94	0.80	0.86	0.64	0.46	0.53	892
Object	test	0.95	0.81	0.88	0.51	0.62	0.56	791
	train	0.96	0.82	0.88	0.51	0.61	0.56	794
Recipient	test	1.00	1.00	1.00	0.52	0.84	0.64	382
	train	1.00	1.00	1.00	0.52	0.83	0.64	384
Predicate	test	1.00	1.00	1.00	0.51	0.64	0.57	888
	train	1.00	1.00	1.00	0.51	0.64	0.57	892
No Role	test	0.97	1.00	0.99	0.92	0.88	0.90	9776
	train	0.97	1.00	0.99	0.91	0.88	0.90	9823

Training and cross-validation classification scores (precision upto 2 decimals) for each output roles predicted by the Word2Vec-ESN classifier in two configuration. Support for each role: actual number of instances, is also shown in last column. Simulation condition: 1000 reservoir neurons, 10-fold cross-validation.

to learn and generalize as well. The classification score during training and testing are lower than that obtained with unscrambled corpus.

Figure 6.2 shows the confusion matrix obtained when using the scrambled corpus as an input to Word2Vec-ESN classifier. It was observed that in configuration-1, the Word2Vec-ESN classifier classifies the argument-predicate pair to the respective roles with high classification scores whereas, in configuration-2, it fails to do so. Comparing the confusion matrix obtained when using unscrambled corpus-462 (see fig. 6.1) and scrambled corpus-462 (see fig. 6.2) it is clearly visible that in configuration-1, the classification scores for each roles is negligibly affected. Whereas in configuration-2, the classification scores of individual roles dropped heavily.

6.4 Experiment-4: Effect of Reservoir size

Word2Vec-θRARes model: Figure 6.3 shows the effect of reservoir size on the cross-validation errors rates (i.e. meaning and sentence error). The meaning and sentence error continuously drops with the increase in reservoir size from 90 to 1098 in both the learning modes. As the reservoir size is further increased from 1098, the meaning error in the SFL mode slightly increases ($\approx 1\%$), whereas it remains unchanged in the SCL mode. On the other hand, the sentence error in

the SFL mode slightly increases whereas in the SCL mode it is negligibly dropped when the reservoir size is further increased from 1776 to 2882 and remains stable after that. Overall, it was observed that both the meaning and sentence cross-validation error rates reduce with increase in reservoir size but asymptotes when the reservoir size is above 1000 on the corpus-462.

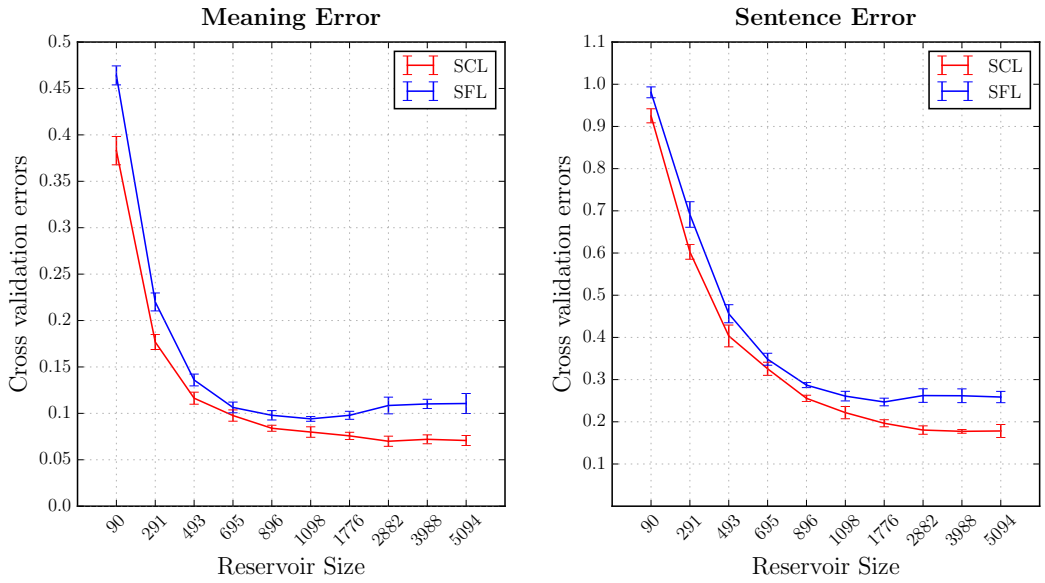

Figure 6.3: Effect of reservoir size on cross-validation errors of Word2Vec-θRARes model.

Word2Vec-ESN classifier: Figure 6.4 shows the change in classification scores of the Word2Vec-ESN classifier with the increase in reservoir size. It can be observed that when using the distributed word vectors (i.e. configuration-1) the classification scores sharply increases when the reservoir size is increased from 50 neurons to 250 neurons. As the reservoir size is further increased from 250 neurons, the classification scores remain stable with the neglible drop of scores. Whereas, in configuration-2 (i.e. when using GF and Localist representation) the classification scores also improves, with the increase in reservoir size from 50 to 250 neurons. As the reservoir size is further increased from 250 neurons, the recall is improved whereas the precision remains almost flat with negligible improvement. The F1-Score is harmonic mean of precision and recall; it was also slightly increased. Even the highest F1-Score of Word2Vec-ESN classifier observed at reservoir size 4500 in configuration-2, is much lower than that of observed at reservoir size 100 ($\approx$ 29%) in configuration-1.

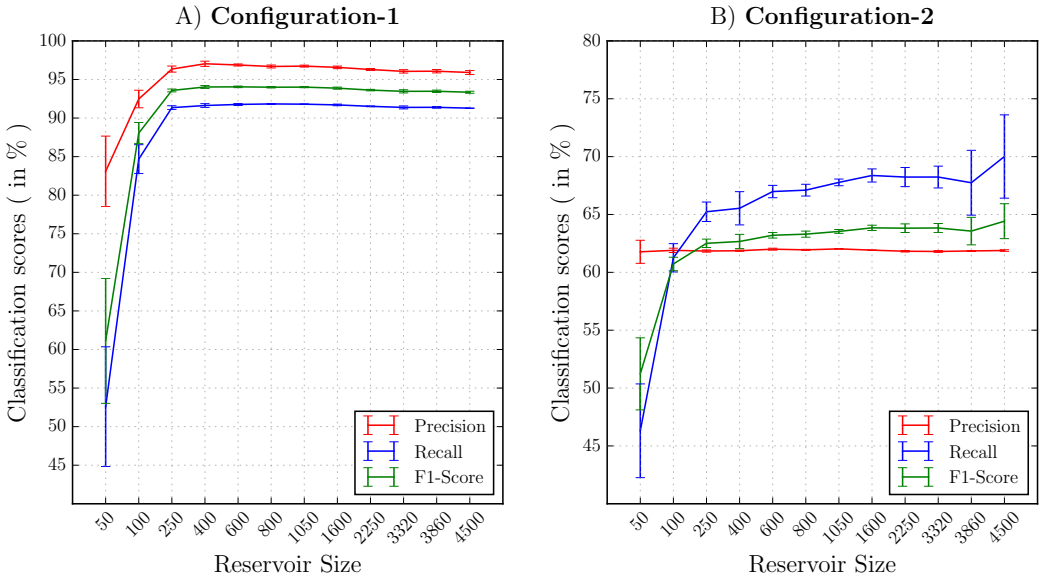

Figure 6.4: Effect of reservoir size on classification scores of Word2Vec-ESN classifier

6.5 Experiment-5: Scalability of the model

Figure 6.5 shows the cross-validation errors rates with respect to corpus size on the Word2Vec-θRARes model. It can be observed that with increase in corpus size from 6% to 50%, the meaning error sharply drops from some 12.18%($\pm$0.19%) to 2.96%($\pm$0.04%) in the SCL mode and from 11.96%($\pm$0.35%) to 4.17%($\pm$0.11%) in the SFL mode. Similarly, the sentence error also decreases from 54.84%($\pm$0.52%) to 17.25%($\pm$0.39%) in SCL and from 54.14%($\pm$1.28%) to 22.41%($\pm$0.57%) in the SFL mode. When the sub-corpora size is 75%, where the model was trained only on 37.5% of corpora size, the model already generalized with 2.72%($\pm$0.03%) meaning error and 15.98%($\pm$0.30%) sentence error in the SCL mode and with 3.95%($\pm$0.11%) meaning error and 21.33%($\pm$0.64%) sentence error in the SFL mode.Although the error rates gradually drops when the corpus size is further increased from 25% to 100% but the improvement rate of errors is very less. Thus the further increase in corpus size will not have much effect on cross-validation error.

Comparing the effect of corpus size on Word2Vec-θRARes model and the θRARes model (see fig. 6.6), it was observed that in SFL mode, with increase in corpus size from 6% to 25%, the meaning error in θRARes model dropped from $\approx$ 17% to $\approx$ 4% and sentence error dropped from $\approx$ 70% to $\approx$ 20%. Whereas in the Word2Vec-θRARes model the meaning error dropped from $\approx$ 12% to $\approx$ 5% and the sentence error dropped from $\approx$ 55% to $\approx$ 26%. Also, with the increase in corpus size further from 25% the cross-validation errors asymptotes in both the models with negligible improvement in error rates. Overall, the improvement in error rates with θRARes model in the SFL mode is higher as compared to Word2Vec-θRARes

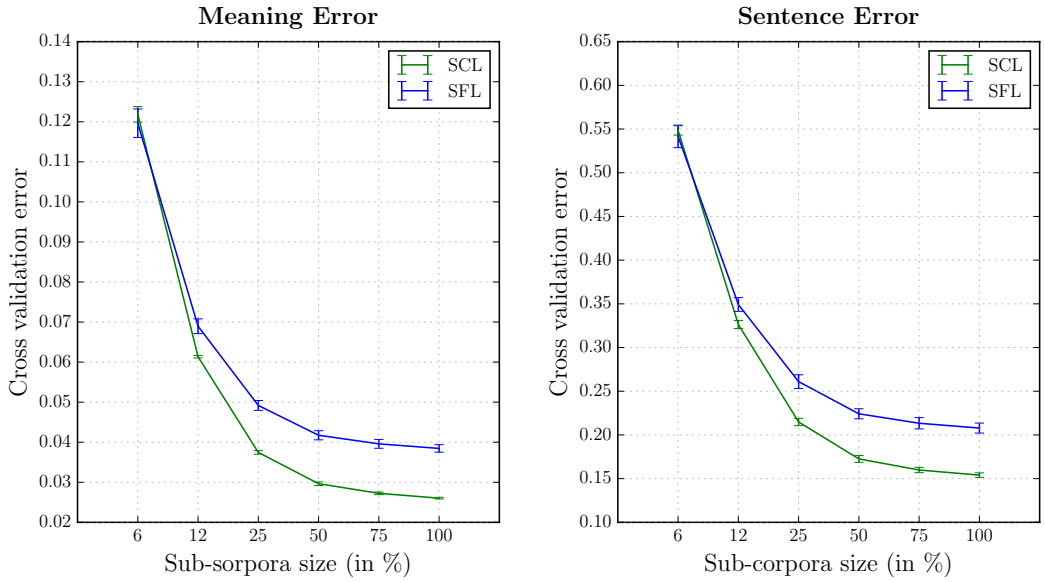

Figure 6.5: **Effect of corpus size on cross validation errors of Word2Vec-SN model in SCL and SFL mode.**

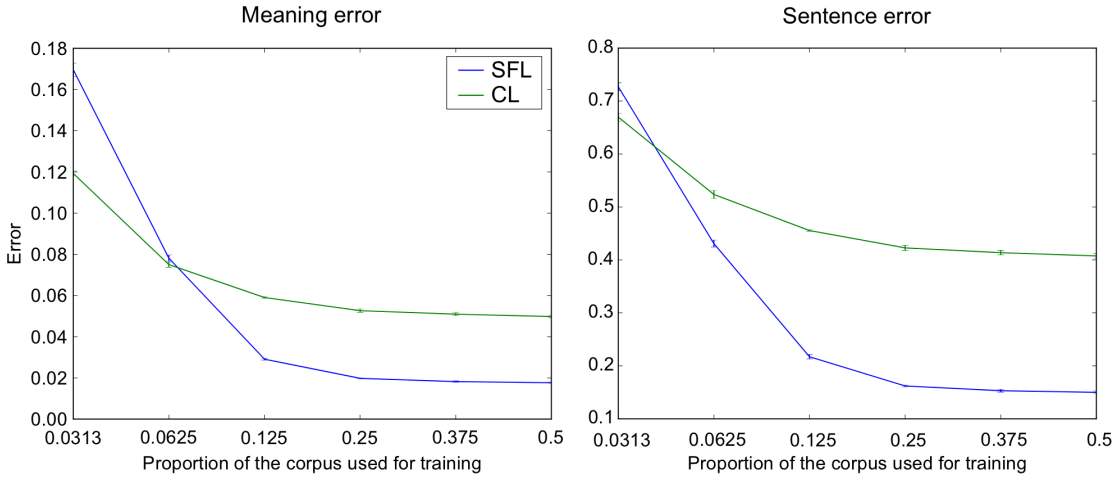

Figure 6.6: Effect of corpus size on cross validation errors using θRARes model in SCL and SFL learning modes.

model.

Comparing the performance of both the models in the SCL mode, it can be seen that the meaning error in Word2Vec-θRARes model continuously drops when the corpus size is further increased from 12% whereas in the θRARes model the decline in meaning error is negligible. The sentence error in θRARes model, drops by a small amount when corpus size is increased from 6% to 25%, but asymptotes as the corpus size is further increased from 25% whereas it gradually drops in Word2Vec-θRARes model. From the lower to the upper limit of corpus size studied, it was observed that the sentence error in θRARes model dropped from $\approx 65\%$ to $\approx 40\%$

whereas in the Word2Vec-θRARes model it dropped from $\approx 55\%$ to $\approx 15\%$. The meaning error, on the other hand, dropped from $\approx 12\%$ to $\approx 5\%$ in the θRARes model whereas, in the Word2Vec-θRARes model it dropped from $\approx 12\%$ to $\approx 2\%$.

Overall in SCL model it can be seen that on the range of sub-corpora size investigated in the experiment, the Word2Vec-θRARes model performed better than θRARes model with all sub-corpora sizes. Another pattern which can be observed is that with all sub-corpora sizes, the Word2Vec-θRARes model performs better in the SCL mode than the SFL mode. Whereas in the θRARes model, with the increase in corpus size the model performs better in SFL model than SCL mode.

6.6 Experiment-6: Generalization on new corpus

Table 6.6 reports the mean sentence error and best generalization error values obtained with Word2Vec-θRARes and θRARes model. The best generalization error here represents the percentage of sentences whose meanings were predicted incorrectly in common within 10 model instances [21]. The Word2Vec-θRARes model with a reservoir of 500 neurons, generalized with a sentence and best error of 42.65% and 26.54% respectively. Whereas with a reservoir of 1000 neurons, the Word2Vec-θRARes model generalized with 40.29% sentence and 25.73% best error. In comparison to the θRARes model, the mean sentence error in the Word2Vec-θRARes model improved by 26.31% and 17.97% with the reservoir of size 500 and 1000 respectively. The best generalization errors also improved by 17.96% and 9.12% with the reservoir size 500 and 1000 respectively.

The Word2Vec-θRARes model generalized better with both the reservoir sizes (500 and 1000) as compared to θRARes model. Notice the improvement in both the model with the increase in reservoir size from 500 to 1000 neurons. It can be seen that the improvement in cross-validation error in Word2Vec-θRARes model is comparatively less than θRARes model. The mean sentence error and best generalization error in θRARes improved by 10.7% and 9.65% respectively. Whereas with Word2Vec-θRARes model the mean sentence error and best generalization error improved by 2.36% and 0.81% respectively with the increase in reservoir size from 500 to 1000 neurons.

Figure 6.7 and 6.8 shows the number of erroneous sentences by number of model instances with reservoir size 500 and 1000 respectively. With a reservoir of 500 neurons (see fig. 6.7), all the ten instances of Word2Vec-θRARes and θRARes model correctly predicted the meaning of 146 and 35 sentences respectively. Whereas the meaning 99 and 166 sentences out of 373 were wrongly predicted by all the ten instances of Word2Vec-θRARes and θRARes model.

With the increase in the reservoir from 500 to 1000 neurons, the number of sentences correctly predicted by all the model instances of Word2Vec-θRARes and θRARes model increased respectively from 146 to 160 and from 35 to 98. Also,

Table 6.6: Generalization error in SCL mode on corpus-373.

Reservoir	Error	Word2Vec-θRARes Model	θRARes Model
500 N	mean(std.)	42.65 ($\pm$ 1.36)	68.96 ($\pm$ 2.03)
	Best	26.54	44.50
1000 N	mean(std.)	40.29 ($\pm$ 1.13)	58.26 ($\pm$ 1.37)
	Best	25.73	34.85

Sentence and best errors (in %) obtained with Word2Vec-θRARes model and θRARes in SCL mode with reservoir of size 500 and 1000 neurons. The results reported are mean and standard deviation of errors obtained from 10 reservoir instances. The best error here means the percentage of sentence errors common within all 10 reservoir instances [21].

the count of sentences where all model instances failed to predict the meaning correctly dropped from 99 to 96 and 166 to 130 for Word2Vec-θRARes model and θRARes model respectively. Notice that with both the reservoir sizes, the number of sentences whose meanings were correctly predicted by all the model instances of Word2Vec-θRARes model is higher than the θRARes model. Whereas the count of sentence wrongly predicted by all model instances are lower in Word2Vec-θRARes model as compared to that of θRARes model.

6.7 Experiment-7: Effect of Word2Vec word dimensions

Word2Vec-θRARes model: Figure 6.9 and 6.10 shows the effect of word vector dimensions on cross-validation errors of Word2Vec-θRARes model in SCL and SFL mode respectively. The corresponding errors are also reported in Table B.2. In the figures, we can see that from lower to the upper limit of the word vector dimensions studied, all the error measures in both the learning modes increased approximately by 2%. Also, in the range 20 to 200 dimensions, the cross-validation errors remained almost equivalent with negligible fluctuations. The minimum errors were obtained with word vectors of 30 dimensions i.e. $ME = 14.23$, $SE = 39.87$ in the SCL mode and $ME = 16.61$, $SE = 43.46$ in the SFL mode. Another interesting pattern which can be observed is that with all the word vector dimensions the model in SCL mode always performed better as compared to SFL mode. The similar relation between SCL and SFL model was also observed in previous experiments as well. Overall we can say that the word vectors dimensions have a negligible effect on the performance of Word2Vec-θRARes model.

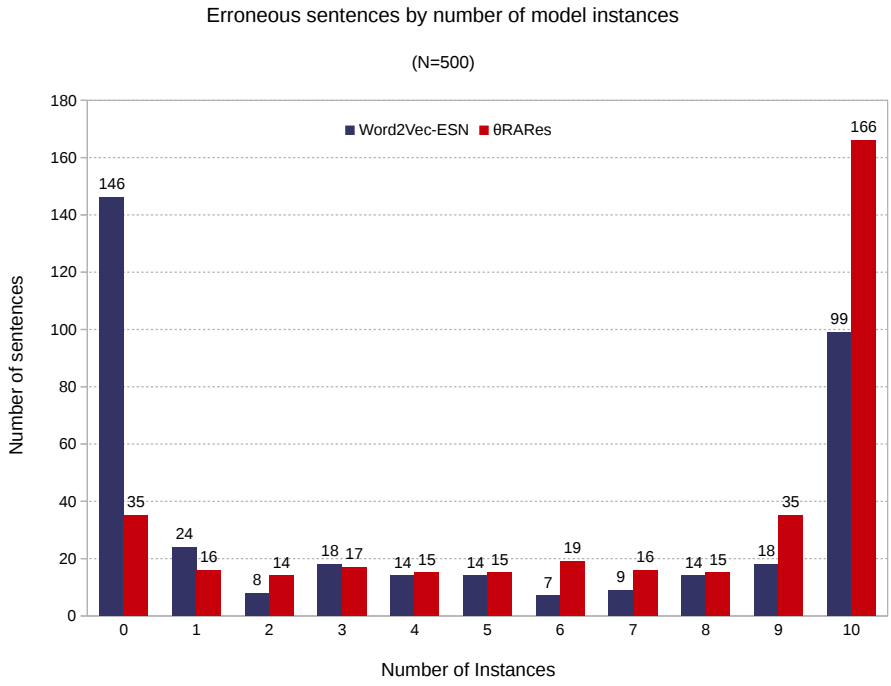

Figure 6.7: Sentence error count by number of instances with reservoir of 500 neurons.

6.8 Neural output activity of the Word2Vec-θRARes model

In the previous experiments, we observed that both the proposed model generalized well and the cross-validation error rates dropped with the increase in corpus size. So, the 45 sentences of corpus-45 were added to corpus-462 to get the resultant corpus have 507 sentences (462 + 45). The readout activations of the Word2Vec-θRARes model were then analyzed for the sentences in corpus-373 and newly formed coprus-507 to have the insight of how the system is processing the distributed word vectors to generate the meaning of the sentences. It was observed that the model is re-analyzing the thematic roles of input sentences across the time. The same behavior was also pointed out by Hinaut et. al [21, 20] with θRARes model.

6.8.1 Output activity for the sentences with toplogically modified coded meaning

Figure 6.11 shows the read out activations of the second noun in the following four sentences across time. Note that the sentences 1 and 2 with active constructions whereas 3 and 4 are passive constructions.

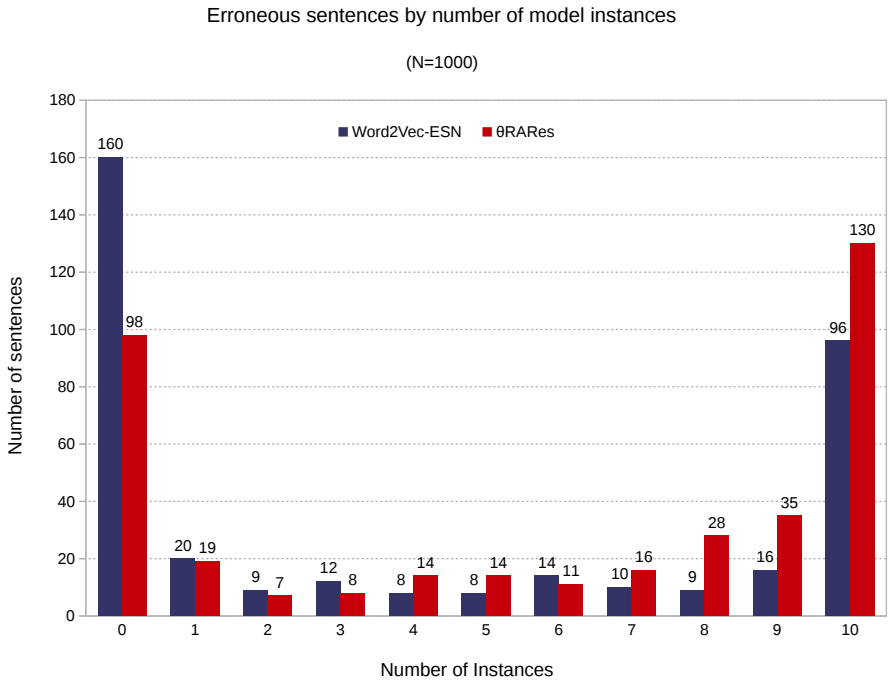

Figure 6.8: Sentence error count by number of model instances with reservoir of 1000 neurons.

1. the man *gave*(V1) the *book*(N2) to the boy.
2. the man *took*(V1) the *ball*(N2) that *hit*(V2) the glass.
3. the boy *caught*(V1) the *ball*(N2) that was *thrown*(V2) by the man.
4. the ball was *pushed*(V1) by the *man*(N2).

Each colored lines in the graph represent the possible thematic roles of the second noun (N2) which can have one of the three possible roles i.e. Agent (A), Object (O) or Recipient (R) with respect to either Verb-1 (V1) or Verb-2 (V2). N2 is marked in red and verbs are marked in green on the x-axis. In the figure the role 'N2-A-V1' can be interpreted as the second noun (N2) is the agent of first verb (V1)..

As all the four sentences start with 'the', activations at this word is same for all four sentences. With the introduction of the first noun ('man') in sentences 1 and 2, the readout activations of role N2 as object of V1 (N2-O-V1) goes above the threshold 0. Thus the model predicts that the N2 could be the object of V1. In sentence 3, with the arrival of the first noun ('boy'), the competition between roles N2-A-V1, N2-A-V2, and N2-O-V2 can be seen, but only the role N2-A-V1 (agent of verb caught) managed to cross the threshold. Whereas, in sentence 4, the activation of role with N2-A-V1 is higher when the first noun ('ball') is encountered, but with the arrival 'was' the role of N2 is changed as recipient of V1 ('pushed').

In sentence 2, with the advent of the V1 ('took') the activation for N2-A-V2

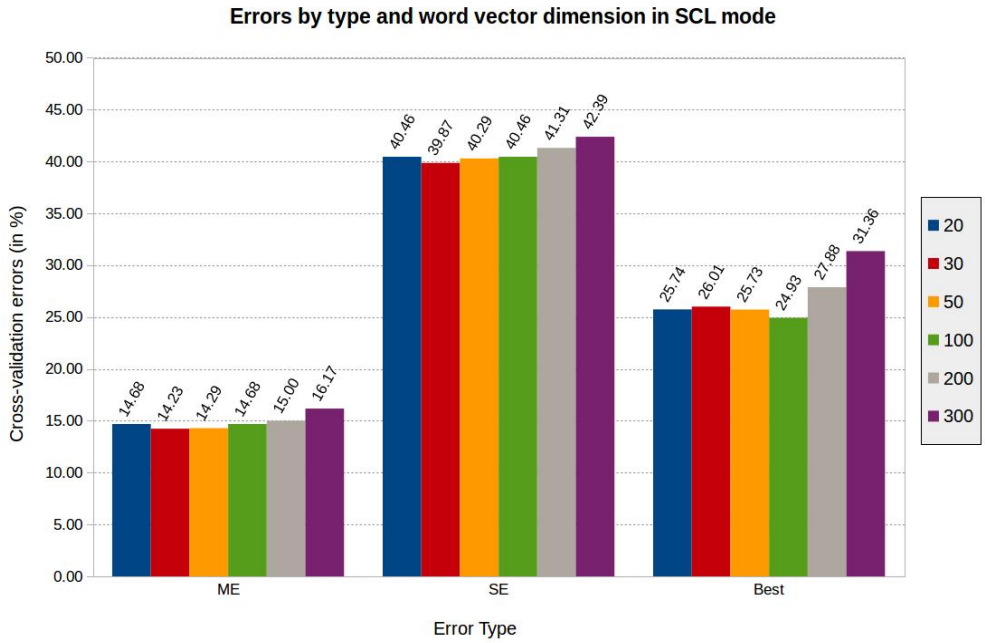

Figure 6.9: **Effect of word vector dimensions on cross validation errors of Word2Vec-θRARes model in SCL mode:** Cross-validation error are given in percentage. Error type includes ME: Meaning Error, SE: Sentence error, Best: Best generalization error.

goes above threshold indicating the presence of V2 ('hit') in the sentence even before the model has encountered the second verb. Whereas in sentence 1, the activation of role N2-A-V1 maintained and no other role managed to cross the threshold with the arrival of V1 ('gave'). In sentence 3 and 4, with the arrival of V1 ('caught' and 'pushed' respectively) one can notice the re-analysis made by the model. In sentence 3, the activation of N2-A-V1 goes below the threshold and takes the new role as the object of both V1 and V2 i.e. N2-O-V1 and N2-O-V2. In sentence 4, the previously predicted role N2-R-V1 goes below threshold with the arrival of V1 ('pushed').

Throughout the rest of sentence 1, the prediction N2-O-V1 is maintained with some minor ups and downs. In sentence 2, the predictions of N2 alternates between O-V1 and A-V2 where the arrival of "the ball" seems to favor O-V1, the arrival of "that" changes the prediction again to A-V2. In both cases, the activation of both the role 0-V1 and A-V2 remains above the threshold. In sentence 3 the model sees both O-V1 and R-V1 as the preferred predictions, both being almost with similar activations and only alternating slightly. In sentence 4, the prediction of role N2 as agent of V1 do not change again throughout the sentence ever since arrival of V1 ('pushed').

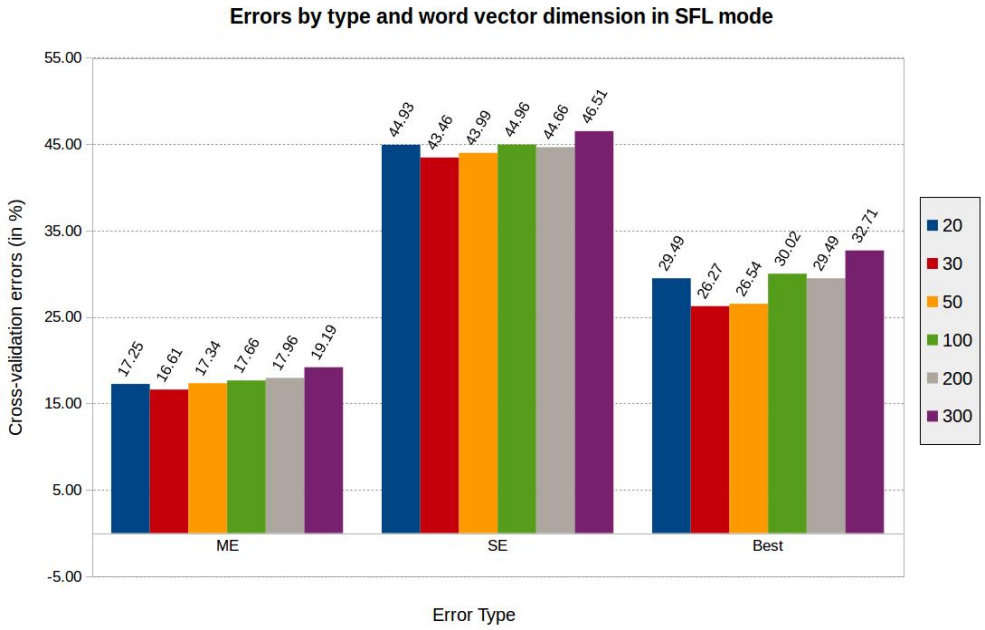

Figure 6.10: **Effect of word vector dimensions on cross validation errors of Word2Vec-θRARes model in SFL mode:** Cross-validation errors are given in percentage. Error type includes ME: Meaning Error, SE: Sentence error, Best: Best generalization error.

Output activity for sentences without toplogically modified coded meaning

Figure 6.12 shows the readout activations of the following two sentences for action-1 and action-2.

(A) *put*(X1) the *cross*(X2) on my *right*(X3)
(B) *push*(X1) the *triangle*(X2) to the *left*(X3) then *push*(X4) the *cross*(X5) to the *left*(X6)

Note that the sentence (A) is a simple sentence with only one action and three semantic words (X1-X3), whereas sentence (B), is a complex sentence with two actions and six semantic words (X1-X6). Each semantic word in the sentences can have a role with respect to an action. A role X1-P1, for example, is interpreted as the first semantic word (X1) is the predicate of action-1. The roles predicate, subject and location are represented by 'P', 'S' and 'L' respectively.

In the first row of Figure 6.12, the readout activity of the model for sentence (A) is shown for each action (2 actions in our case). One can see that for action-1, the model correctly predicts the meaning *put(cross, right)*, whereas for the action-2, at the end of sentence all the neural activations go below the threshold 0. Thus, this indicates the presence of only action-1 in the sentence. Also notice the online

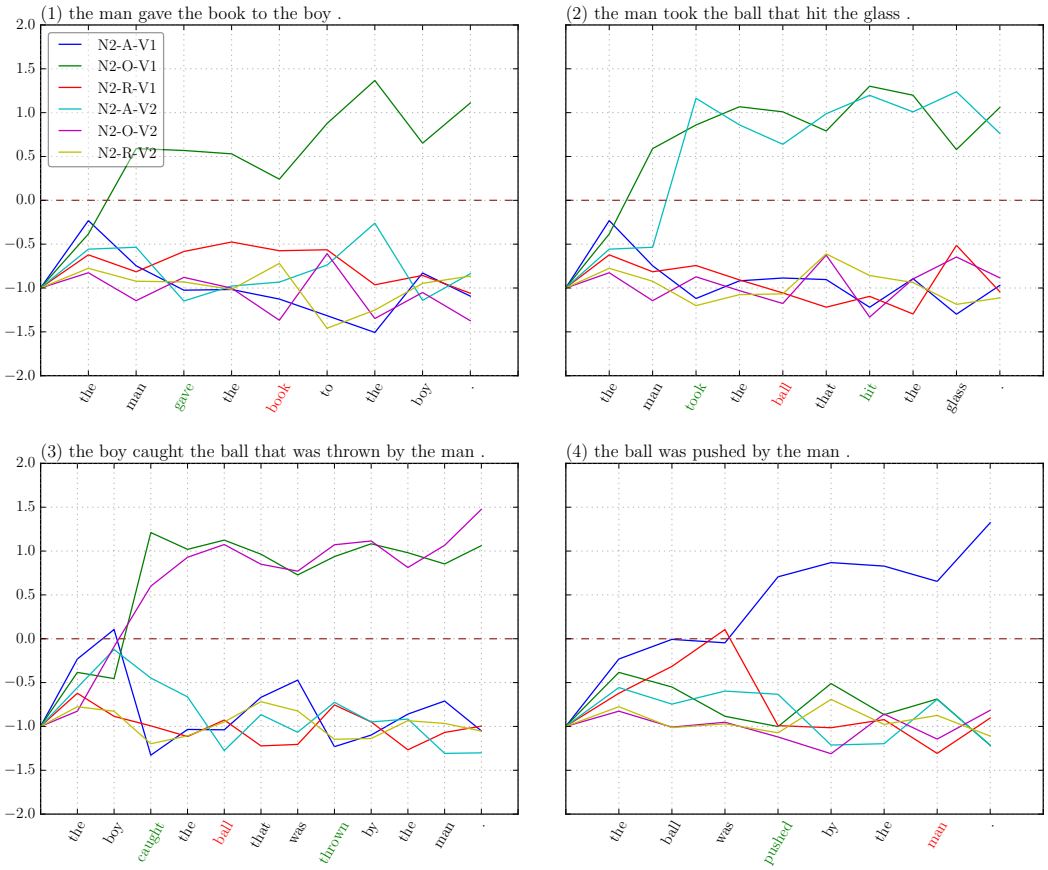

Figure 6.11: **Online re-analysis of activation by Word2Vec-θRARes Language model:** Each coloured line shows the thematic role of Noun-2 (tick marked in red) with respect to Verb-1 and Verb-2 (ticks shown in green).

re-analysis behavior of the model, where for the action-2, the activation of role (X1-L2) goes above threshold 0 at the beginning of the sentence but later drop below 0 at the end of the sentence.

The second row of Figure 6.12, illustrate the readout activity of sentence (B) for each action. The model correctly predicts the thematic roles of constituent semantic words (X1 to X6) for both the actions. For the action-1 the model makes early predictions of roles i.e. *push (triangle, left)*, whereas, for action-2, the activation of correct roles are reinforced when the model encounters the second predicate (X4 = 'push') in the sentence. Thus the model predicts the meaning of sentence for the second action as *push (cross, left)*.

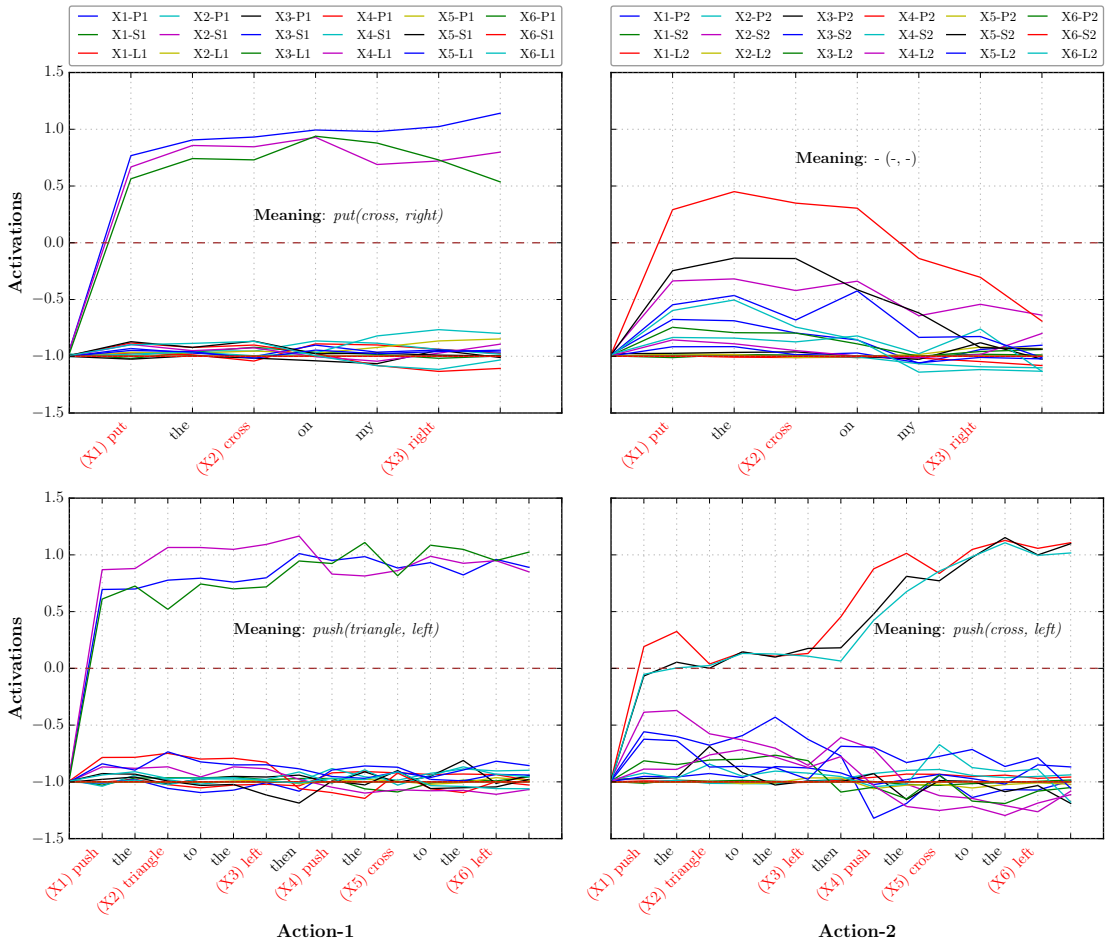

Figure 6.12: **Readout activity of Word2Vec-θRARes model for a simple and complex sentence from corpus-373:**

6.9 Evaluating Word2Vec-θRARes model performance

In the previous section, we saw the online re-analysis of sentence meanings by Word2Vec-θRARes model. Previously, we also saw the performance of Word2Vec-θRARes model in numbers. In this section, we will see that for what type of the sentences the proposed model performed better than θRARes model on TRA task and vice-versa. This analysis was done on results obtained in Experiment-6 using corpus-373.

While analyzing the sentences whose meanings were correctly predicted by all ten instances of Word2Vec-θRARes model and incorrectly predicted by θRARes model, a significant pattern was identified. Almost 85% of the sentences wrongly predicted by all instances of θRARes model were either two action commands or redundant commands (see fig. 3.3). For example "push the circle in the middle and put it in the right" and "push the circle to the middle and then put it on the

right".

Table 6.7 and Table 6.8, reports the sentences whose meanings were predicted wrongly at least by one instance of θRARes model and correctly predicted by all the instances of the Word2Vec-θRARes model. It was observed θRARes model failed to predict the meaning of most of the sentences in which the color of the objects to be moved is specified. In other words, the model failed to predict the meaning os sentences where of adjectives like *'red'* or *'blue'* are used with semantic words (see table 6.7). Similarly, sentences in Table6.8, includes anaphoric reference word *'it'* (sentences 1-3), pointing to a semantic word within the discourse of sentence . Also, there are instructions which specify an action to be repeated using the anaphoric words 'twice' or 'two times' (sentences 4-7). Such sentences were also predicted incorrectly by at least one instance of θRARes model. Whereas, the proposed Word2Vec-θRARes model, correctly predicts the meanings these sentences. θRARes model also failed to predict the meaning of sentences where the reference location was given with respect to the person instructing the robot (sentences 1, 8, 9), whereas Word2Vec-θRARes model predicted them correctly.

Table 6.7: Example sentences correctly predicted by all 10 instances of Word2Vec-θRARes model and mispredicted atleast once with θRARes model.

S.No	**Sentences**	**Actual Meaning Predicate(Object,Location)**	
		Action-1	***Action-2***
1	point the red cross and point the circle	point(cross, -)	point(circle, -)
2	put the red cross to the left and hit the blue circle	put(cross, left)	hit(circle, -)
3	grasp the red cross and then point the circle	grasp(cross, -)	point(circle, -)
4	after putting the red cross to the left please hit the blue circle	putting(cross, left)	hit(circle, -)
5	please grasp the red cross	grasp(cross, -)	-

Table 6.8: My caption

S.No	**Sentences**	**Actual Meaning Predicate(Object,Location)**	
		Action-1	***Action-2***
1	before grasping the triangle on my left point at it.	point (triangle, -)	grasping(triangle,-)
2	before grasping the triangle point at it.	point (triangle, -)	grasping(triangle,-)
3	before hitting the triangle point at it.	point (triangle, -)	hitting (triangle,-)
4	grasp circle two times	grasp(circle, -)	grasp(circle, -)
5	point cross two times	point(cross,-)	point(cross, -)
6	hit twice the blue circle	hit(circle, -)	hit(circle, -)
7	grasp the circle twice	grasp(circle,-)	grasp(circle, -)
8	point the circle on my left	point (circle, -)	-
9	push the cross on my left and then grasp the circle	push (cross, left)	grasp (circle, -)

So far, we have seen the upside of Word2Vec-θRARes model over θRARes model, but there are cases where Word2Vec-θRARes model fails as well and are

reported in Table 6.9. It was observed that in most of the cases, the meaning of the sentences with complex structures where the first action to be performed is specified after the second action, using words 'after having' are predicted wrongly by both the models (sentences 2-4). Also, the Word2Vec-θRARes model fails to predict the meaning of grammatically incorrect sentences in most of the cases e.g. sentence 1. It was also noticed that use of word contractions e.g. 'youve', 'weve' etc. in the sentences also leads to an error in meaning prediction by Word2Vec ESN model.

Table 6.9: My caption

Sentences	**Sentences**	**Actual Meaning Predicate(Object,Location)**	
		Action-1	***Action-2***
1	in left push the triangle and the cross	push(triangle, left)	push(cross, left)
2	grasp the circle but before put the cross on the left	put(cross, left)	grasp(circle,-)
3	touch the cross after having touch the triangle	touch(triangle,-)	touch(cross,-)
4	put the triangle to the left after having touched it	touched (triangle,-)	put triangle left
5	touch the circle after youve pointed the triangle	pointed(triangle,-)	touch(circle,-)
6	put the cross over to the left and then once youve done that grasp the circle	put(cross, left)	grasp(circle,-)

Chapter 7

Conclusion And Future Work

Previously, we saw the θRARes model which process the sentences transformed to grammatical form and using localist word representations to solve the TRA task [20]. We have also seen the proposed neural-inspired Word2Vec-θRARes language model and the experiments performed using it. In this chapter, we finally conclude this research, in two sections first, by summarizing the findings from the outcome of experiments performed, second, the possible direction to future work identified during the development of this research.

7.1 Conclusion

This thesis started with a goal to test the hypothesis that distributed word representation in combination with a recurrent neural network can be used to achieve better performance on TRA task. Thus, to test this hypothesis, Word2Vec-θRARes language model and Word2Vec-ESN classifier was proposed. The experiments were performed using the proposed Word2Vec-θRARes model and Word2Vec-ESN classifier. The outcomes of the experiments have led to the conclusion that follows.

7.1.1 Generalization: Performance on unseen sentences

One basic conclusion from this research is that the proposed Word2Vec-θRARes model using distributed word embeddings performs better than θRARes model which process the sentences transformed in the grammatical form using localist word vectors.

Performance on limited number of sentences: It was shown in Experiment-1, that the Word2Vec-θRARes model learned the sentences thoroughly when trained on limited sentences but fails to generalize on unseen sentences. Whereas, the θRARes model was able to generalize better on the small corpus because of the generalization imposed in the data by replacing the semantic words with 'SW' token while pre-processing. This generalization leads us to a conclusion that θRARes model performs better on small corpus whereas Word2Vec-θRARes fails to do so.

Performance on corpus-462 and corpus-90582:

Performance on corpus-462 and corpus-90582: Recall that the extended corpus-462 and corpus-90582 contains the sentences with the verb inflections (i.e. -ed, -ing, -s). The θRARes model process the sentences transformed in the grammatical form using the localist word vectors. Thus, the input layer contains a unit for each closed class words, a unit to encode semantic word ('SW' token) and additional units to encode the verb inflections [20]. Whereas, for the Word2Vec-θRARes model, the verbs with inflections in the corpora were replaced with the corresponding verb conjugations while preprocessing. The Word2Vec-θRARes model uses the topologically modified coded meaning for experiments. Use of verb inflections as an input in θRARes model and topologically modified coded meaning with Word2Vec-θRARes model does not suffice the requirement of an end-to-end system. Both the θRARes and Word2Vec-θRARes model has to depend on a parser to extract the verb inflections and verbs respectively in a sentence. Thus the performance of the model becomes dependent on the accuracy of the parser. Assuming that the parser is 100 % accurate, the Word2Vec-θRARes model generalized on both the corpora but performed much better in the SCL mode as compared to that of the SFL mode whereas, it is vice-versa in θRARes model. The Word2Vec-θRARes model outperforms the θRARes model in the SCL mode. The word embeddings learned from the nearby words and the capability of ESN to model the long-dependencies in the sentences can be credited for such behavior. In the SFL mode, the performance of both the Word2Vec-θRARes θRARes model remained almost equivalent. So, it can be concluded that when both the models are dependent on a parser, the Word2Vec-θRARes model performs better than θRARes model.

The Word2Vec-ESN classifier also generalizes with high classification scores, when the raw sentences are processed with distributed words embeddings. The classifier also generalizes with low classification scores, when the sentences transformed in grammatical form and encoded using localist vector representation are used for processing. Overall the results indicate that the distributed word embeddings improve the performance of the Word2Vec-ESN classifier over localist word vectors.

Performance on coprus-373: As corpus-373 does not contain verb inflections, thus the θRARes model process the sentences without using the input units for verbs inflections. Also, the Word2Vec-θRARes model does not use topologically modified coding. This makes both the models independent of the parser and thus the performance of both the models can be compared. As shown in the Experiment-6, the Word2Vec-θRARes model outperforms the θRARes model. Hence, it can be concluded the semantic and syntactic information captured in distributed word vectors helps the Word2Vec-θRARes model to enhance the performance on TRA task.

7.1.2 Effect of reservoir and corpus sizes

It was also shown that the generalization ability of the Word2Vec-θRARes model on TRA task also increases with increase in reservoir size but asymptotes at a point. With the increase in reservoir size, the Word2Vec-θRARes model always generalized better in SCL mode than SFL mode. Also, the Word2Vec-θRARes model requires substantial corpus for learning thematic roles and generalize well on unseen sentences. However, the generalization ability becomes stable after extending the corpus size to a certain limit. Overall, the conclusion which can be drawn from this is that with large corpus and reservoir size the Word2Vec-θRARes model generalizes better.

The Word2Vec-ESN classifier, on the other hand, achieved the high classification score even with the smaller reservoir size of 250 neurons. The performance of the model does not deviate much with the further increase in reservoir size. Thus the model can be used with small reservoir size which makes it computationally cheaper for TRA task.

7.1.3 Structure of corpus

The results of Experiments 2 and 3, shows that the Word2Vec-θRARes model generalizes well in both the SCL and SFL mode when the inherent grammatical structure is present in the sentences but fails to generalize when the word order in the sentences are scrambled. Thus we can conclude that the Word2Vec-θRARes model is not generalizing on any inconsistent regularity in the corpus but, instead, exploiting the inherent grammatical structure of the sentences to learn and generalize.

On the other hand, the Word2Vec-ESN classifier generalized with high classification scores with the scrambled corpus in configuration-1 (with distributed word embeddings). However, this behavior is not surprising at all. The answer to this behavior lies in the way the Word2Vec-ESN classifier is trained. Recall that unlike Word2Vec-θRARes model the Word2Vec-ESN classifier is trained to classify an argument-predicate pair to a role. So the model learns the mapping between the argument-predicate pairs and the corresponding roles. So even if the word order is changed, the model utilizes the semantic and syntactic information and also the information captured about context words in word embeddings, to classify the argument-predicate pairs correctly. The conclusion which can be made from here is that the Word2Vec-ESN classifier can perform better with distributed embeddings in the cases where the sentences are not completely grammatically correct.

In the configuration-2 (with localist word vectors), the Word2Vec-ESN classifier failed to generalize on the scrambled corpus. The reason is the transformation of sentences to grammatical form. The use of same 'SW' token for all semantic words and encoding the words in localist fashion confuses the model to assign the appropriate thematic role to the input argument-predicate pair thus leading to failure in generalization.

7.1.4 Robustness of the model

It was shown in Experiment-6, that the Word2Vec-θRARes model gracefully generalized well on corpus-373, using the same model parameters learned from corpus-462 with the topologically modified coded meaning. Thus, it can also be concluded that the model parameters are robust enough to generalize on new corpora for TRA task. However, there is a possibility to optimize the performance further on corpus-373 with Word2Vec-θRARes model by optimizing the ESN parameters on this corpus.

7.1.5 Dimension of word vectors

Mikolov et al. [34, 33] showed that the higher dimensional word embeddings performed better on word analogy task as compared to word vectors with lower dimensions. Whereas we have seen in the result of Experiment-7, that the dimensions of distributed word vectors, do not affect the performance of Word2Vec-θRARes model significantly on TRA task. However the larger word vector dimensions increase the computation cost, but the performance gain is negligible. So, following the Occam's razor principle which states that "It is vain to do with more what can be done with fewer" [18], it can be concluded that the smaller word vectors should be preferred over larger ones with the Word2Vec-θRARes model for TRA task.

7.1.6 Online re-anlysis of sentence meanings

In addition to the generalization on unseen sentences, it was also shown that the proposed model also do the re-analysis of the sentence meanings upon arrival of each word. The read-out activity of the model changes each time a new word is input to the model. The read-out activations at any time step can be interpreted as estimated prediction of the meaning of sentence for the input given so far [20]. However, this behavior is not something new, the online re-analysis of sentence meanings was also be observed with θRARes model. However, the use of word embeddings learned from the context words and possibly encapsulating the information about neighboring words enables the Word2Vec-θRARes model to make very earlier prediction of sentences meanings. This behavior seems to be natural in humans when a sentence is being heard. A person listening to a sentence starts predicting the meaning of a sentence as soon as he/she listen to first few words by making assumptions, gained from the previous experiences. This advantage of early prediction can be useful for Human-Robot Interaction (HRI). A robot can start making actions, using the output activations, even before the completion of the sentence [21].

The Word2Vec-ESN classifier considers the TRA task as a classification problem. The training objective of the classifier is to classify the argument-predicate pairs to the thematic roles. Thus, unlike the Word2Vec-θRARes model, the Word2Vec-ESN classifier was not teacher forced with the meaning of the whole sentence during training but instead with the thematic role of the current argument-

predicate pair. Thus it is not possible for the Word2Vec-ESN classifier to do the online re-analysis of sentence meaning.

7.1.7 Word2Vec-θRARes Vs Word2Vec-ESN classifier

We have seen that the Word2Vec-θRARes model and Word2Vec-ESN classifier, process the sentences differently and are evaluated using different metrics. As the performance of both the models is evaluated with different metrics, it will not be fair to claim which one of them performs better. However, it can be argued that the both the models perform better when distributed word embeddings are used over localist word vectors. For the TRA task, both the models could be employed. Each model has its advantages and limitations which are as follows:

Input coding: The Word2Vec-θRARes model process a sentence word by word. Thus the model takes a word as an input at each time step. Whereas, the Word2Vec-ESN classifier takes an argument-predicate pair as input. To identify the predicates (verbs) in a sentence, the Word2Vec-ESN classifier has to depend on a syntactic parser. Hence the performance of the Word2Vec-ESN classifier also depends on the accuracy of the parser. This is one of the limitations of Word2Vec-θRARes model Word2Vec-ESN classifier.

Output coding: The output units of the Word2Vec-θRARes model encodes the thematic roles of the semantic words. Thus to encode the thematic roles for all possible semantic words in any sentence, the maximum number of semantic words a sentence can have in the corpus should be known in prior. Also, the number of output units increases with increase in semantic words and the thematic roles a semantic word can have. The increase in the number of output units also increase the number of learnable reservoir-to-outputs weights (W^{out}) and hence the computational cost also increases. The number of output units in the Word2Vec-ESN classifier, on the other hand, unlike Word2Vec-θRARes model, remains independent of the number of semantic words a sentence can have. The number of output units is always equal the number of thematic roles which a word can have. Thus the limited number of output units makes the Word2Vec-ESN classifier computationally cheap.

Reservoir size: The Word2Vec-θRARes model requires a bigger reservoir for generalization, whereas the Word2Vec-ESN classifier generalizes with a small reservoir size. Generalization with the small reservoir size also makes the Word2Vec-ESN classifier computationally cheap and hence a preferable choice over Word2Vec-θRARes model.

Online re-analysis of sentence meaning: As discussed earlier, the Word2Vec-θRARes have the capability to make online re-analysis of sentence meaning across

time. In some cases, the model also predicts the meaning of sentence even before the completion of sentence. This advantage of online re-analysis and early prediction is not possible with Word2Vec-ESN classifier.

7.2 Future Work

For this work, we currently used the combination of word2vec model and ESN. The ESN has an advantage of modeling sequential data. Thus the sequential and temporal aspect of a sentence is taken into account in this study for thematic role assignment. But the dependencies between the thematic roles of semantic words in the sentences were not taken into consideration for learning. To model the conditional probability distribution of the thematic roles of semantic words, Conditional Random Fields (CRF); a log-linear model; could be used [48]. CRFs has been one of the most successful approaches used earlier as well for classification and sequential data labeling tasks [55, 12]. Thus the Word2Vec-θRARes model, proposed in this study, could be used with an additional CRF unit to model the temporal dependencies between the input sentences conditional on the corresponding thematic roles. Doing so allows the resulting model to capture the concealed temporal dynamics present in the sentences [12].

Distributed word vectors obtained by training the word2vec model on one language corpus (say English) can also be translated to get the most similar word in any target language. This translation is achieved by linearly projecting the word vector of source language on target language [35]. Thus the Word2Vec-θRARes language model can also be investigated further for multiple language acquisition [22].

In this research, the proposed Word2Vec-ESN classifier makes an assumption the syntactic parser used to identify verbs is 100% accurate. It would be intersting to explore the behavior of the Word2Vec-ESN classifier with the most accurate syntactic parser like stanford parser [46], Charniak parser [11], Bikel parser [8] or Berkley parser [40].

Appendix A

Nomenclature

Appendix B

Complete Simulation Results

Table B.1: Effect of sub-corpora size on Word2Vec-ESN model in different learning modes.

sub-corpora size		Word2Vec-ESN			
		SCL mode		SFL mode	
		ME	SE	ME	SE
5 %	mean	12.18	54.84	11.96	54.15
	std	0.19	0.52	0.35	1.28
12 %	mean	6.13	32.63	6.89	34.92
	std	0.03	0.46	0.18	0.78
25 %	mean	3.74	21.48	4.91	26.10
	std	0.04	0.42	0.12	0.78
50 %	mean	2.96	17.25	4.17	22.41
	std	0.04	0.39	0.11	0.57
75 %	mean	2.72	15.98	3.95	21.33
	std	0.03	0.30	0.11	0.64
100 %	mean	2.60	15.40	3.84	20.78
	std	0.02	0.25	0.09	0.57

Meaning (ME) and Sentence error (SE) in different learning modes with Word2Vec-ESN model. The errors are given in percentage. The sub-corpora (first column) is randomly sampled from corpus-90582. SCL: Sentence Continuous Learning; SFL: Sentence Final Learning; std: standard deviations. Simulations were done with 5 model instance each having a reservoir of 1000 neurons.

Table B.2: Effect of word vector dimensions on Word2Vec-ESN model in different learning modes.

		Word2Vec-ESN language model					
		SCL mode			SFL mode		
		ME	SE	Best	ME	SE	Best
20	mean	14.68	40.46	25.74	17.25	44.93	29.49
	std	0.58	1.58	-	0.76	1.36	-
30	mean	14.23	39.87	26.01	16.61	43.46	26.27
	std	0.53	0.57	-	0.67	0.70	-
50	mean	14.29	40.29	25.73	17.34	43.99	26.54
	std	0.34	1.13	-	0.36	1.38	-
100	mean	14.68	40.46	24.93	17.66	44.96	30.02
	std	0.58	1.58	-	0.55	1.66	-
200	mean	15.00	41.31	27.88	17.96	44.66	29.49
	std	0.73	1.55	-	0.63	1.19	-
300	mean	16.17	42.39	31.36	19.19	46.51	32.71
	std	0.75	1.46	-	0.64	1.34	-

Meaning (ME), Sentence Error (SE) and Best value for different word vector dimensions in different learning modes with Word2Vec-ESN model. The errors are given in percentage. SCL: Sentence Continuous Learning; SFL: Sentence Final Learning; std: standard deviations. Simulations were done using corpus-373, with 10 model instance each having a reservoir of 1000 neurons. LoO cross-validation approach was used to evaluate the model.

Bibliography

[1] Glossary of terms. *Mach. Learn.*, 30(2-3):271–274, February 1998.

[2] Gerry Altmann and Richard Shillcock. *Cognitive models of speech processing: The second Sperlonga meeting*, chapter 9, page 165. Erlbaum, 1993.

[3] Mohamed Aly. Survey on multiclass classification methods. *Neural Netw*, pages 1–9, 2005.

[4] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247, 2014.

[5] Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. Textual inference and meaning representation in human robot interaction. In *Joint Symposium on Semantic Processing*, 2013.

[6] Elizabeth Bates, Sandra McNew, Brian MacWhinney, Antonella Devescovi, and Stan Smith. Functional constraints on sentence processing: A cross-linguistic study. *Cognition*, 11(3):245–299, 1982.

[7] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[8] Daniel M Bikel. A distributional analysis of a lexicalized statistical parsing mode. In *EMNLP*, pages 182–189, 2004.

[9] Xavier Carreras and Lluís Màrquez. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 152–164, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[10] Xavier Carreras and Llus Mrquez. Introduction to the conll-2004 shared task: Semantic role labeling, 2004.

[11] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics, 2000.

[12] Sotirios P Chatzis and Yiannis Demiris. The echo state conditional random field model for sequential data modeling. *Expert Systems with Applications*, 39(11):10303–10309, 2012.

[13] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[14] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[15] Peter Ford Dominey and Thomas Voegtlin. Learning word meaning and grammatical constructions from narrated video events. In *Proceedings of the HLT-NAACL 2003 workshop on Learning word meaning from non-linguistic data-Volume 6*, pages 38–45. Association for Computational Linguistics, 2003.

[16] Kenji Doya. Bifurcations in the learning of recurrent neural networks 3. *learning (RTRL)*, 3:17, 1992.

[17] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[18] J. Franklin. *The Science of Conjecture: Evidence and Probability Before Pascal*. A Johns Hopkins paperback. Johns Hopkins University Press, 2002.

[19] Adele E Goldberg. *Constructions: A construction grammar approach to argument structure*. University of Chicago Press, 1995.

[20] Xavier Hinaut and Peter Ford Dominey. Real-time parallel processing of grammatical structure in the fronto-striatal system: A recurrent network simulation study using reservoir computing. *PLoS ONE*, 8(2):1–18, 02 2013.

[21] Xavier Hinaut, Maxime Petit, Gregoire Pointeau, and Peter Ford Dominey. Exploring the acquisition and production of grammatical constructions through human-robot interaction with echo state networks. *Frontiers in Neurorobotics*, 8:16, 2014.

[22] Xavier Hinaut, Johannes Twiefel, Maxime Petit, France Bron, Peter Dominey, and Stefan Wermter. A recurrent neural network for multiple language acquisition: Starting with english and french. In *Proc. of the NIPS 2015 workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*, 2015.

[23] Xavier Hinaut and Stefan Wermter. An incremental approach to language acquisition: Thematic role assignment with echo state networks. In *International Conference on Artificial Neural Networks*, pages 33–40. Springer, 2014.

[24] Google Inc. Vector Representations of Words. `https://www.tensorflow.org/versions/r0.10/tutorials/word2vec/index.html`. Accessed: 2016-09-05.

[25] H. Jaeger. Echo state network. 2(9):2330, 2007. revision 151757.

[26] Herbert Jaeger. A tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the" echo state network" approach.

[27] Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34, 2001.

[28] Herbert Jaeger. Adaptive nonlinear system identification with echo state networks. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 609–616. MIT Press, 2003.

[29] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352, 2007.

[30] Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 181–184, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[31] Ding Liu and Daniel Gildea. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 716–724. Association for Computational Linguistics, 2010.

[32] Mantas Lukoševičius. *A Practical Guide to Applying Echo State Networks*, pages 659–686. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[33] T Mikolov and J Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.

[34] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[35] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[36] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2009.

[37] Arzucan Özgür, Levent Özgür, and Tunga Güngör. *Text Categorization with Class-Based and Corpus-Based Keyword Selection*, pages 606–615. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[38] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.

[39] Jacob Persson, Richard Johansson, and Pierre Nugues. Text categorization using predicate–argument structures. *Proe. the*, 1:142–149, 2008.

[40] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics, 2006.

[41] Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. Semantic role chunking combining complementary syntactic views. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 217–220, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[42] Sameer S Pradhan, Wayne Ward, Kadri Hacioglu, James H Martin, and Daniel Jurafsky. Shallow semantic parsing using support vector machines. In *HLT-NAACL*, pages 233–240, 2004.

[43] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[44] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[45] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[46] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *ACL (1)*, pages 455–465, 2013.

[47] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.*, 45(4):427–437, July 2009.

[48] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *arXiv preprint arXiv:1011.4088*, 2010.

[49] Ciza Thomas and N. Balakrishnan. Improvement in minority attack detection with skewness in network traffic, 2008.

[50] Geoffrey Zweig Tomas Mikolov, Scott Wen-tau Yih. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics, May 2013.

[51] Matthew H. Tong, Adam D. Bickett, Eric M. Christiansen, and Garrison W. Cottrell. 2007 special issue: Learning grammatical structure with echo state networks. *Neural Networks*, 20(3):424–432, 2007.

[52] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[53] Francisco J Valverde-Albacete and Carmen Peláez-Moreno. 100% classification accuracy considered harmful: The normalized information transfer factor explains the accuracy paradox. *PloS one*, 9(1):e84217, 2014.

[54] Xiaofeng Wang, Matthew S Gerber, and Donald E Brown. Automatic crime prediction using events extracted from twitter posts. In *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*, pages 231–238. Springer, 2012.

[55] Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2015.

Erklärung der Urheberschaft

Ich versichere an Eides statt, dass ich die Master Thesis im Studiengang Intelligent Adaptive Systems selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum Unterschrift

Erklärung zur Veröffentlichung

Ich erkläre mein Einverständnis mit der Einstellung dieser Master Thesis in den Bestand der Bibliothek.

Ort, Datum Unterschrift