



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
14.10.2016

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 language. More interestingly, they learn it within the same neural computing substrate. But, how 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 have been proposed. Most of the existing models are either using syntactic parse trees to create handcrafted features or using localist vector representation of words to process the sentences. Hinaut et al. [20] proposed the θ RARes (Thematic Role Assignment Reservoir) neural language acquisition model, for the Thematic Role Assignment (TRA) task. The model learns the thematic roles, i.e., “Who did what to whom”, purely from the grammatical structure of the sentences. The semantic information was removed from the sentences by replacing semantic words with a unique token (‘SW’). The model processes the transformed sentences considering the words as discrete atomic symbols i.e. using localist word representation. The localist representation of the words does not carry any semantic and syntactic information about the words. This thesis proposes an end-to-end neural language model, Word2Vec- θ RARes, in an extension of θ RARes model. In addition to utilizing syntactic information and modeling temporal aspect of the input sentences, the proposed model also takes into account the semantics of the words. The model contains a Word2Vec unit to generate the distributed vector representation of words which captures the semantic and syntactic relationships. The model also contains a Recurrent Neural Network namely, Echo state Network (ESN), with fixed random connections, which is capable of modeling the sequential data. Thus, the model receives a raw sentence as an input which is processed word by word across time, and the meaning of the sentence is generated as an output. The results of the experiments conducted using the Word2Vec- θ RARes model shows that the exposure of semantic and syntactic information to the ESN increases the performance on the TRA task. Apart from this, the Word2Vec- θ RARes model also re-analyse the meaning of the sentence on arrival of words. Also, in some cases, the model correctly predicts the meaning of the sentences even before the sentences are finished.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Previous Work | 1 |
| 1.2 | Motivation and Hypothesis | 2 |
| 1.3 | Proposed Models | 3 |
| 1.4 | Scope and Novelty | 4 |
| 1.5 | Outline | 5 |
| 2 | Basics of Word2Vec Model and Echo State Network | 7 |
| 2.1 | Word2Vec Model | 7 |
| 2.1.1 | CBOW model | 8 |
| 2.1.2 | Skip-Gram model | 9 |
| 2.1.3 | Properties of Word2Vec embeddings | 11 |
| 2.2 | Echo State Network (ESN) | 13 |
| 2.2.1 | ESN Architecture | 13 |
| 2.2.2 | Training ESN | 15 |
| 3 | Related Work: θRARes Model and Its Limitation | 17 |
| 3.1 | Overview of θ RARes Model | 17 |
| 3.1.1 | Limitations of θ RARes model | 20 |
| 3.2 | Research Hypothesis | 23 |
| 4 | Approaching Word2Vec-θRARes Language Model | 25 |
| 4.1 | Word2Vec- θ RARes Language Model | 25 |
| 4.1.1 | Training Word2Vec- θ RARes model | 26 |
| 4.1.2 | Decoding output | 28 |
| 4.1.3 | Evaluation metrics | 29 |
| 4.2 | Word2Vec-ESN Classifier | 29 |
| 4.2.1 | Training Word2Vec-ESN classifier | 31 |
| 4.2.2 | Decoding output | 31 |
| 4.2.3 | Evaluation metrics | 31 |
| 5 | Material and Methods | 35 |
| 5.1 | Corpora and Pre-processing | 35 |
| 5.1.1 | Corpora for the TRA task | 35 |
| 5.1.2 | Corpus for training Word2Vec model | 37 |

| | | |
|---------------------|--|-----------|
| 5.2 | Experimental Setup | 38 |
| 5.2.1 | Obtaining distributed word embeddings | 38 |
| 5.2.2 | ESN reservoir initialization | 39 |
| 5.2.3 | Learning ESN parameters | 40 |
| 5.2.4 | Input and Output coding | 41 |
| 6 | Experiments and Results | 43 |
| 6.1 | Experiment-1: Performance on Small Corpus | 43 |
| 6.2 | Experiment-2: Generalization Capabilities | 44 |
| 6.3 | Experiment-3: Effect of Corpus Structure | 49 |
| 6.4 | Experiment-4: Effect of Reservoir Size | 51 |
| 6.5 | Experiment-5: Scalability of the Model | 53 |
| 6.6 | Experiment-6: Generalization on New Corpus | 56 |
| 6.7 | Experiment-7: Effect of Word2Vec Word Dimensions | 58 |
| 6.8 | Neural Output Activity of the Word2Vec- θ RARes Model | 60 |
| 6.8.1 | Output activity for sentences with topologically modified coded meaning | 61 |
| 6.8.2 | Output activity for sentences without topologically modified coded meaning | 62 |
| 6.9 | Comparative Analysis of Sentences Meaning | 64 |
| 7 | Conclusion And Future Work | 67 |
| 7.1 | Conclusion | 67 |
| 7.1.1 | Generalization: Performance on unseen sentences | 67 |
| 7.1.2 | Effect of reservoir and corpus size | 69 |
| 7.1.3 | Structure of corpus | 69 |
| 7.1.4 | Robustness of the model | 70 |
| 7.1.5 | Dimension of word vectors | 70 |
| 7.1.6 | Online re-analysis of sentence meanings | 70 |
| 7.1.7 | Word2Vec- θ RARes vs Word2Vec-ESN classifier | 71 |
| 7.2 | Future Work | 72 |
| A | Complete Simulation Results | 75 |
| Bibliography | | 81 |

List of Figures

| | | |
|------|--|----|
| 2.1 | The CBOW model. | 8 |
| 2.2 | The Skip-gram model. | 10 |
| 2.3 | Semantic regularities in distributed word embeddings. | 12 |
| 2.4 | Clustering of words in word embedding space. | 12 |
| 2.5 | Language Translation property of distributed word vectors. | 13 |
| 2.6 | Architecture of classical ESN. | 14 |
| 3.1 | Schematic characterization of TRA in grammatical construction. | 18 |
| 3.2 | Functional organization of θ RARes language model. | 19 |
| 3.3 | Relation between sentence surface form and meanings. | 20 |
| 4.1 | Architecture of Word2Vec- θ RARes model. | 25 |
| 4.2 | Neural comprehension of the Word2Vec- θ RARes model. | 27 |
| 4.3 | Neural comprehension of the Word2Vec- θ RARes model with topologically modified coded meaning. | 28 |
| 4.4 | Functional organization of Word2Vec-ESN classifier for the TRA task. | 30 |
| 6.1 | Normalized confusion matrix for unscrambled corpus-462. | 48 |
| 6.2 | Normalized confusion matrix for scrambled corpus-462. | 49 |
| 6.3 | Effect of reservoir size on Word2Vec- θ RARes model. | 52 |
| 6.4 | Effect of reservoir size on Word2Vec-ESN classifier. | 53 |
| 6.5 | Effect of corpus size on Word2Vec- θ RARes model. | 54 |
| 6.6 | Effect of corporous size on θ RARes model. | 55 |
| 6.7 | Sentence error count by number of model instances with reservoir size 500. | 57 |
| 6.8 | Sentence error count by number of model instances with reservoir size 1000. | 58 |
| 6.9 | Effect of word vector dimensions on Word2Vec- θ RARes model in SCL mode. | 59 |
| 6.10 | Effect of word vector dimensions on Word2Vec- θ RARes model in SFL mode. | 60 |
| 6.11 | Readout activity of Word2Vec- θ RARes model for a sentence with topologically modified coded meaning. | 61 |
| 6.12 | Readout activity of Word2Vec- θ RARes model for a simple and complex sentence from corpus-373. | 63 |

List of Tables

| | | |
|-----|--|----|
| 3.1 | Transformation of a sentence to abstract grammatical form and encoding it using localist word vectors. | 19 |
| 4.1 | Confusion matrix to evaluate the Word2Vec-ESN classifier on the TRA task. | 32 |
| 6.1 | Cross-Validation errors with Word2Vec- θ RARes on limited number of sentences. | 44 |
| 6.2 | Generalization score with Word2Vec-ESN classifier on limited number of sentences. | 45 |
| 6.3 | Meaning and Sentence error in the SCL and SFL modes. | 46 |
| 6.4 | Classification scores of the Word2Vec-ESN classifier on unscrambled and scrambled corpus-462 in two configurations. | 47 |
| 6.5 | Training and testing classification scores for individual roles when using Word2Vec-ESN classifier in two different configurations. | 50 |
| 6.6 | Word2Vec- θ RARes model generalizing on new corpus | 57 |
| 6.7 | Example sentences correctly predicted by all 10 instances of Word2Vec- θ RARes model and mispredicted atleast once by θ RARes model. | 65 |
| 6.8 | Example sentences for which the Word2Vec- θ RARes model failed to predict the meaning correctly. | 65 |
| A.1 | Effect of reservoir size on Word2Vec- θ RARes model in different learning modes. | 76 |
| A.2 | Effect of reservoir size on Word2Vec-ESN classifier in two different configurations. | 77 |
| A.3 | Effect of sub-corpora size on Word2Vec- θ RARes model in different learning modes. | 78 |
| A.4 | Effect of word vector dimensions on Word2Vec- θ RARes model in different learning modes. | 79 |

List of Tables

Abbreviations

| | | | |
|-------------|------------------------------|-----------|-----------------|
| CBOW | Continuous Bag Of Word | TP | True Positive |
| TRA | Thematic Role Assignment | FP | False Positive |
| SRL | Semantic Role Labelling | FN | False Negative |
| SVM | Support Vector Machine | SR | Spectral Radius |
| RNN | Recurrent Neural Network | IS | Input Scaling |
| ESN | Echo State Network | LR | Leak Rate |
| ESP | Echo State Property | ME | Meaning Error |
| CNN | Convolutional Neural Network | SE | Sentence Error |
| CRF | Conditional Random Field | Pr | Precision |
| NLP | Natural Language Processing | Re | Recall |
| SCL | Sentence Continuous Learning | F1 | F1-Score |
| SFL | Sentence Final Learning | A | Agent |
| LoO | Leave One Out | P | Predicate |
| HRI | Human Robot Interaction | O | Object |
| SW | Semantic Word | R | Recipient |
| SG | Skip-Gram | | |

Chapter 1

Introduction

Thematic Role Assignment (TRA) is a supervised learning problem which aims to identify the events and its participants in a sentence to determine “*Who did what to whom*”. In other words assigning roles to the words (arguments) in a sentence with respect to a verb (predicate). The roles typically include 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 verb ‘*gave*’ and object of verb ‘*hit*’. In Natural Language Processing (NLP), the similar problem is studied under the name of Semantic Role Labelling (SRL). Hence, SRL is a form of simplistic semantic parsing which aims to determine the argument-predicate structure for each verb (predicate) in a given sentence [56]. 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 [55], document categorization [40], 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, 43, 42]. These systems are based on the pre-defined feature templates derived from syntactic information obtained by parsing and producing parse trees of the sentences in a training corpus. However, in a study it was found that the use of syntactic parser certainly leads to the degradation of thematic roles predictions [42]. This is because the parsers themselves are not always completely accurate. 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 [56].

In order to evade manual feature engineering, the SRL task was additionally attempted using the 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). However, the model was less successful as CNNs cannot employ long-term dependencies within a sentence since it can only take into account the words in limited context [56]. However, to increase the model’s performance, they also resorted to using syntactic features by using parse trees of Charniak parser [11].

Recurrent Neural Network (RNN) has also been used for a wide range of NLP tasks. Recently Echo State Network (ESN), a variant of RNN, have also been used for such tasks. The tasks performed were diverse: from predicting the next word given the previous words to learning grammatical structures and also recently on the TRA task [52]. A RNN makes use of sequential information and acts as a memory unit to capture the information processed in the past [17]. The ESNs, on the other hand, have several advantages over simple RNNs. First, while processing long sequences, the gradient parameter do not vanish vanishes or explodes in ESNs [33] whereas RNNs suffers from vanishing gradient problem [7]. Second, unlike simple RNNs, ESNs are computationally cheap as the recurrent layer (reservoir) in an ESN is randomly initialized and only the connections from recurrent layer to read-out layers are learned [28, 33]. These advantages of ESN over RNN make it an excellent choice to be used for the TRA task.

Hinaut et al. [20] proposed a generic neural network architecture, the θ RARes model, using RNN based on the reservoir computing approach, namely Echo State Network to solve the TRA task. The proposed architecture models the language acquisition in the brain and provides a robust and scalable implementation on the 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’ in this case). The transformed sentences encoded in localist fashion are then input to the model sequentially, word by word, across time along with the coded meaning (i.e. thematic roles of the semantic words) of the input sentence for training. The model learns the thematic roles of all the semantic words in the input sentences during training. During testing, the model predicts the coded meaning of the previously unseen sentences. More details about the θ RARes model will be discussed later in chapter 3.

1.2 Motivation and Hypothesis

As described earlier, the θ RARes model does not process the raw sentences but processes the sentences transformed to abstract grammatical form. This transformation of sentences by replacing the semantic words with a ‘SW’ token makes it possible to train the θ RARes model on a small corpus as the ‘SW’ token can be replaced with different semantic words (nouns, verbs, etc.) to form several sentences [20]. However, 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. Like many

other traditional NLP systems, Hinaut et al. [20] also treated the words as discrete atomic symbols and used localist vector representation of words as an input to the θ RARes model. Treating each word as a discrete symbol does not provide any relational information to the model which may exist between the words. For example, if the words ‘*pink*’ and ‘*red*’ are represented using localist representation with vectors [1,0] and [0,1] respectively, then the semantic relationship between these two words (i.e. both are colors) is lost [24]. Thus the transformation of a sentence to grammatical form and encoding it using localist representation, deprive a system from the semantic information present in words. This could make it difficult for the model to learn the thematic roles of sentences. The possible limitations of this model are discussed in more detail in Chapter 3.

Mikolov et al. [34] proposed a neural model, widely known as the Word2Vec model, to learn the distributed representations of words. Word2Vec model learns high quality, low-dimensional vector representation of words from a large corpus in an unsupervised way. The model learns the word embeddings by taking into account the context (neighboring) words. The resulting word vectors of the model encapsulate the syntactic and semantic information about the words. The obtained word embeddings also capture several language regularities and patterns [36, 34] and can be observed by performing the linear operations on the word vectors. For example, $\text{vector}(\text{'king'}) - \text{vector}(\text{'man'}) + \text{vector}(\text{'woman'}) \approx \text{vector}(\text{'queen'})$. Unlike other neural network models [39, 13, 53, 37] for obtaining word embeddings, training Word2Vec is fast, computationally cheap and efficient [34]. Training of Word2Vec model and properties of resulting distributed word embeddings will be discussed later in more detail in Chapter 2.

This work is motivated by the limitations of localist word representations and transformation of raw sentences into its abstract grammatical form described above. To overcome these limitations, this thesis hypothesizes that processing raw sentences and using the distributed representations of words as an input to ESN could possibly enhance the performance of the TRA task. The distributed word vectors are learned from the context words and encapsulate the semantic and syntactic relationship of the words [34]. Thus the exposure of this syntactic and semantic information of words to the ESN could allow the network to learn these dynamics and hence improve the performance on the TRA task.

1.3 Proposed Models

To test the hypothesis, this thesis proposes an end-to-end neural language model, the Word2Vec- θ RARes model, in extension to θ RARes model for thematic role assignment. The Word2Vec- θ RARes 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- θ RARes model for the 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 words and generate the distributed word embeddings. The

generated word vectors can then be used by ESN for learning the thematic roles of the input sentences. Unlike the θ RARes model, the Word2Vec- θ RARes model process the raw sentences. Also, the distributed word vectors are used over localist word representation as an input to ESN.

Apart from the Word2Vec- θ RARes model, this thesis also proposes a Word2Vec-ESN classifier. The classifier only differs from the Word2Vec- θ RARes model in the way the sentences are processed, and the results are evaluated. The input and the output of the classifier, differ from the Word2Vec- θ RARes model. Like Word2Vec- θ RARes, the classifier also processes the sentences sequentially. The input feature to the Word2Vec-ESN classifier is the current word (argument) and the predicate with respect to which it is processed. Thus a sentence is processed as many times as there are verbs in a sentence. Unlike the θ RARes model where output units encode the possible thematic roles of all semantic words in the input sentences, the output units of Word2Vec-ESN classifier encode the possible role (e.g. Predicate, Agent, Object, Recipient and No Role) of the input argument-predicate pairs. For the evaluation of Word2Vec-ESN classifier, the metric proposed for CoNLL-2004 [10] and CoNLL-2005 [9] SRL task is used. Both the Word2Vec- θ RARes language model and the Word2Vec-ESN classifier are discussed in more detail in Chapter 4.

Using the proposed models, this thesis will address the following questions:

1. **Generalization:** How good a model is in predicting the thematic roles of the sentences not used for training?
2. **Model Size:** How the performance of a model is impacted by the size of the model (i.e. reservoir size)?
3. **Scalability:** How the performance of a model is affected by the increase in corpus size?
4. **Robustness:** Can a model be used to perform the TRA task on the corpus not used for learning the model's parameters?
5. **Distributed Vector Dimensions:** What will be the effect on the performance of a model when the distributed word vectors of different dimensions are used?
6. **Real-time Processing:** How the readout activations of the model are affected while parsing a sentence?

1.4 Scope and Novelty

There are several neural network based models to obtain the word embeddings [39, 13, 53, 37] but a systematic comparison of them for the 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.

1.5 Outline

This thesis is organized as follows. Chapter 2, describes the basics of the Word2Vec model, the Echo State Network model and how they are trained. The chapter also describes the properties of word vectors obtained by training the Word2Vec model and also the control parameters of ESN. Chapter 3 looks upon related work with the primary focus on the θ RARes language model, its limitations, and revisits the motivation and hypothesis for the current work. Subsequently, chapter 4 proposes the Word2Vec- θ RARes model and the Word2Vec-ESN classifier. This chapter also describes the implementation and the metrics used to evaluate the performance of proposed models. Chapter 5 first describes the corpora used to perform the experiments and then describes the necessary initialization of the proposed models before using them for the TRA task. In chapter 6, the experiments performed, and results obtained using the proposed model on the TRA task are described. In this chapter, we also see the comparison of the results obtained using the proposed models and the θ RARes model. Finally, chapter 7 conclude this study and also makes 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, the basics of Word2Vec model, the flavors in which the model exist, the training of the Word2Vec model and the properties of the word embeddings learned by the model. In the second section, the basics of Echo State Network and how it is trained is described.

2.1 Word2Vec Model

Word2Vec model is a neural probabilistic language model proposed by Mikolov et al. [34]. 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 from the context words (nearby words). Thus training of Word2Vec is considered to be an unsupervised task [44]. The model takes a large text data, to generate the distributed embeddings of the words present in the text and also preserves 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 [35]. For learning the word embeddings, the Word2Vec model exists in two neural architectures, the Continuous Bag Of Word (CBOW) and Skip-Gram (SG) [35, 34]. Figure 2.1 and 2.2 shows the neural architecture of CBOW and SG models respectively. Both the CBOW and SG model have three neural layers, i.e., an input layer, a hidden layer and an output layer. Although, both the models are architecturally similar but have different training objectives. The next sections, provide a

brief overview of CBOW model and SG model. However, in this work skip-gram model will be used because it was shown that it generate better word embeddings as compared to CBOW [34].

2.1.1 CBOW model

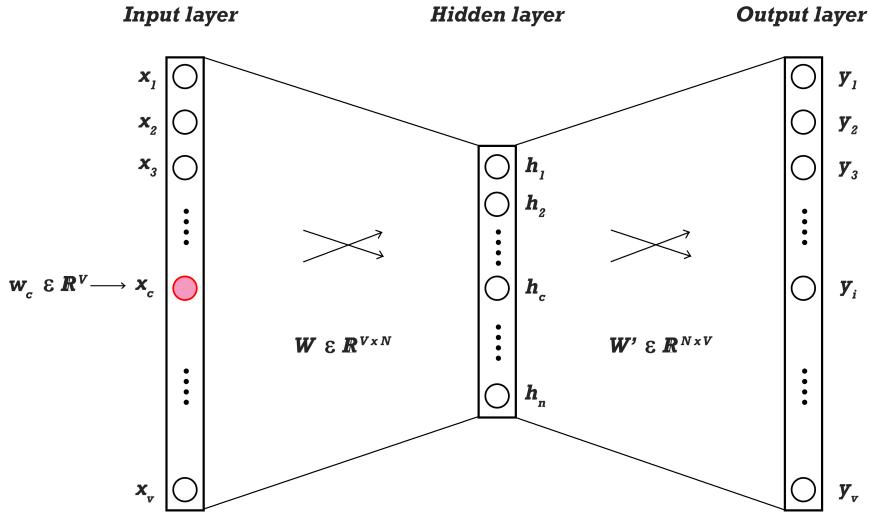


Figure 2.1: The CBOW model: The objective of CBOW model is to predict the target word based on the context words (or neighboring words). Given a vocabulary V , and desired vector dimension N , the context words are input to the model (in this case only one, w_c) using localist representation. Only one vector element corresponding to the input word is one (x_c , shown in red) and all other elements are 0. The model outputs the probability for each word in vocabulary, which is then maximized for the actual target word during training. Adapted from [45].

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 . The hidden layer is of size N which equals the dimension of the desired word embeddings. The neurons in both the adjacent layers i.e. input and output layer 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, $w_c = [x_1, \dots, x_c, \dots, x_V]$ where only one unit x_c at index c will be 1 out of V units and all other units will be 0 [45]. The activation of the hidden layer is then given by:

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

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 . Equation 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.

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 [45].

$$\begin{aligned} u &= W'^T \cdot h \\ &= W'^T \cdot v_c \end{aligned} \quad (2.1.2)$$

$$u_i = W_i'^T \cdot v_c \quad (2.1.3)$$

where u_i gives the score of i^{th} word, w_i , for $i = 1, 2, \dots, V$. $W' \in \mathbb{R}^{N \times V}$ 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 the output neurons with softmax activation function [45]. Thus aliasing W_i' as v_i' , we get output probabilities as:

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

where y_i is the probability of i^{th} word (w_i) given the context word (w_c). The training objective is then achieved by maximizing the log likelihood of the actual target word (w_t) given the context word (w_c) [45]. So the cost functions can be written as:

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

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

$$h = \frac{1}{K} \cdot (v_{c_1} + v_{c_2} + \dots + v_{c_K}) \quad (2.1.6)$$

where k is the size of the context window. This equation averages the vector embeddings of all the context words [45].

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 a word that is good in predicting the context words [24, 34]. Thus for a given sequence of words $\{w_1, \dots, w_V\}$, the objective is to maximize the average log probability.

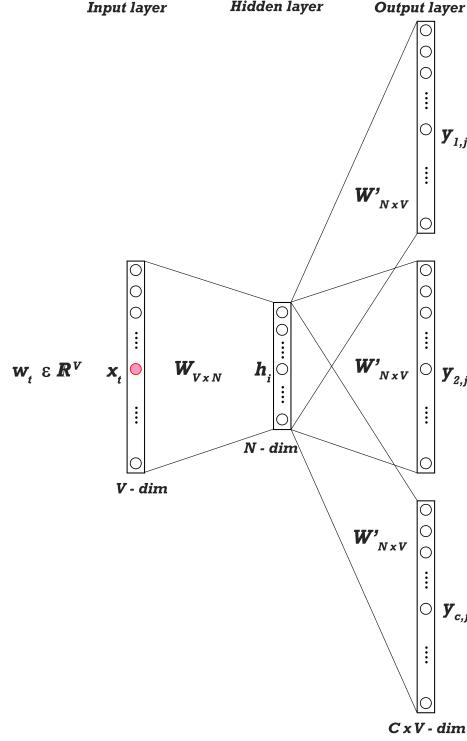


Figure 2.2: **The skip-gram model:** The objective of Skip-gram model is the reverse of CBOW. It predicts the context words from a target word. Given a vocabulary V , and desired vector dimension N , The target word (w_t) is input to the model using localist representation where only one vector element corresponding to the input word is 1 (x_t , shown in red) and all other are 0. The model maximizes the probability of the context words during training. Adapted from [45].

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

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:

$$p(w_{t+j}|w_t) = \frac{\exp(v'_{t+j} \cdot v_t)}{\sum_{w \in V} \exp(v'_w \cdot v_t)} \quad (2.1.8)$$

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 and the gradient is calculated using the Back Propagation rule [45, 36]. Calculating the full softmax is computationally expensive as it needs to compute and normalize the probability for every other word w in the vocabulary V (see eqn. 2.1.4 and 2.1.8) for a given input word (w_c for CBOW or

w_t for skip-gram) at every training step [34]. Thus negative sampling approach was proposed as an alternative for learning the word embeddings.

Skip-gram with negative sampling

For learning word features, full probabilistic models were not required. Therefore, the skip-gram negative sampling approach is used for approximation of word features. This technique treats the feature learning as a binary classification (logistics regression) problem [34, 24]. The model is thus trained to distinguish the target word from k imaginary noise words w_{noise} , in the same context. Hence, the log probability $p(w_{t+j}|w_t)$ in equation 2.1.8 is now approximated by:

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

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 to improve the quality of words vectors, the most frequently occurring words in the training corpus can be subsampled [34]. A large training corpus usually contains common words like “a”, “the”, “in” etc. which occurs million times. The model benefits 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 [34]. So in order to reduce the effect of most frequent words, each word (w_i) in the vocabulary of the training corpus was discarded with probability:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \quad (2.1.10)$$

where $f(w_i)$ is the frequency of word w_i and t is a threshold. Using this formula, the frequencies of the words are preserved while all the words with a frequency greater than t are subsampled [34].

2.1.3 Properties of Word2Vec embeddings

The Word2Vec model is simple in its architecture and easy to train. It also produces the word embeddings which surprisingly encodes several linguistic regularities and patterns [36, 34]. Moreover, it is astonishing because the network was not explicitly trained for these linguistic properties. 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 [34]. For example, $vector('King') - vector('Queen') \approx vector('man') - vector('woman')$, $vector('apples') - vector('apple') \approx vector('cars') - vector('car')$, $vector('walking') - vector('walked') \approx vector('swimming') - vector('swam')$ (see fig. 2.3).

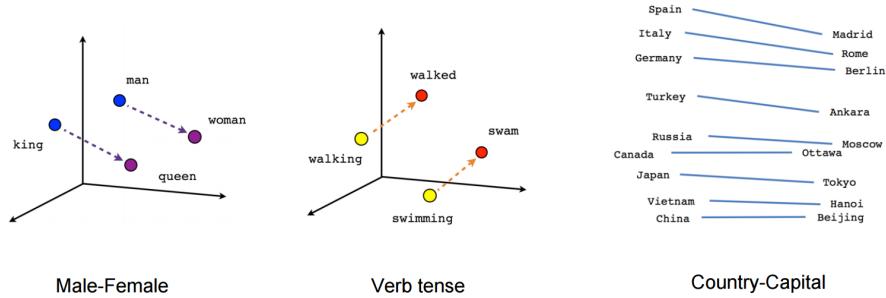


Figure 2.3: Semantic regularities in Word2Vec word embeddings [24].

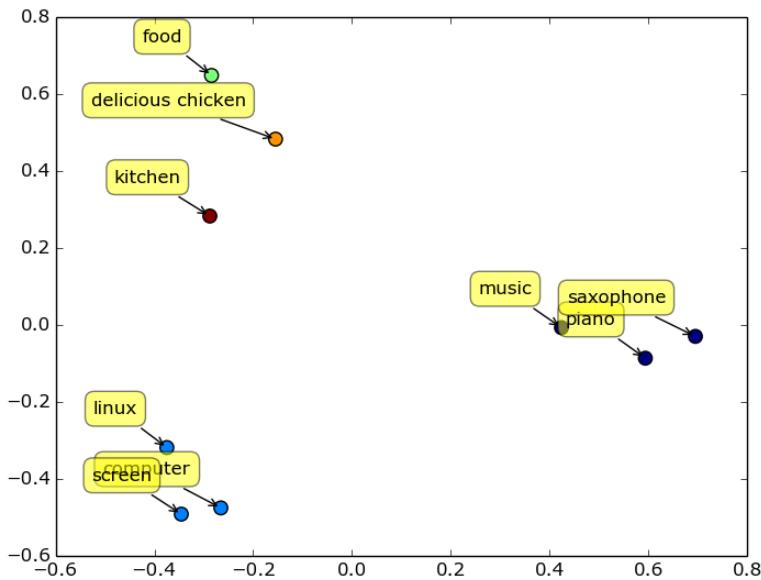


Figure 2.4: Clustering of words in word embedding space: The figure shows the word clustering property of distributed word embeddings obtained by projecting the word vectors on two dimesional space using PCA. Figure adapted from: <https://github.com/aubry74/visual-word2vec>.

Another interesting property of distributed word embeddings is that the semantically related words are placed close to each other in the word embedding space [34], thus forming clusters of semantically related words (see fig. 2.4). Additionally, the word embeddings of similar words in different languages have the same geometrical arrangement in the word embedding space of respective language (see fig. 2.5). Therefore, it is also possible to learn the linear mapping between different word embedding space by vector rotation and scaling [36]. Several other regularities can also be observed by performing the basic linear operation on word-embeddings [34].

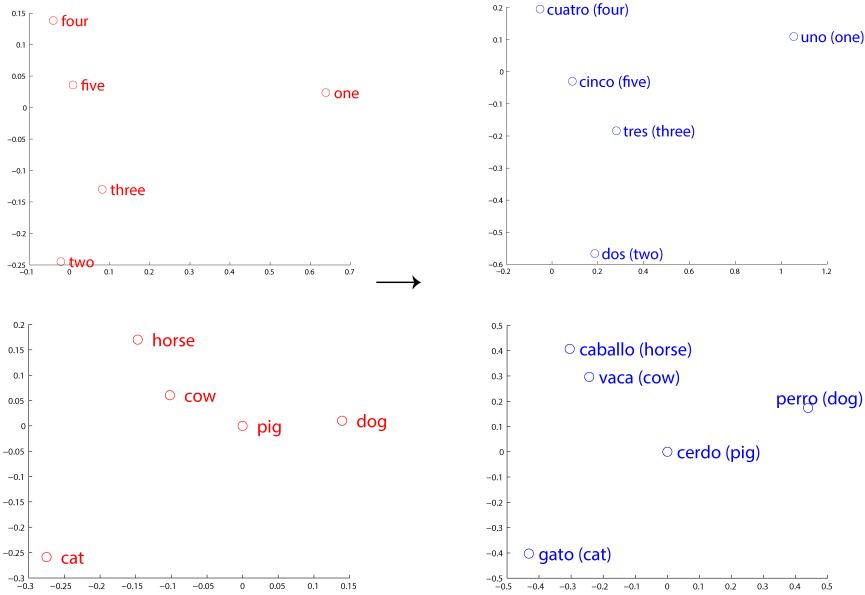


Figure 2.5: Language Translation property of distributed word vectors:
 The figure shows the distributed word vectors in English (left) and Spanish (right), projected in 2-D space using PCA. Notice that both the languages have the same geometrical arrangement of similar words in word embedding space. Thus, the word vectors in the English language can be rotated and scaled to obtain the word embeddings of similar words in Spanish. Figure adapted from [36].

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 resemble the learning mechanism in biological brains closely. ESN is found to be computationally simple and inexpensive to process the temporal or sequential data. The main idea of ESN is to operate a random, large, and crucially, a 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) [46] but, instead the linear regression is used. The reason for this choice is that the use of BP makes the training of RNN non-converging, slow, and computationally expensive [16]. Whereas, the training using linear regression is faster and computationally inexpensive [33].

2.2.1 ESN Architecture

ESN is a surprisingly efficient variant for training the RNNs. Figure 2.6 shows the basic architecture of an ESN. In a standard RNN, all the weights are required

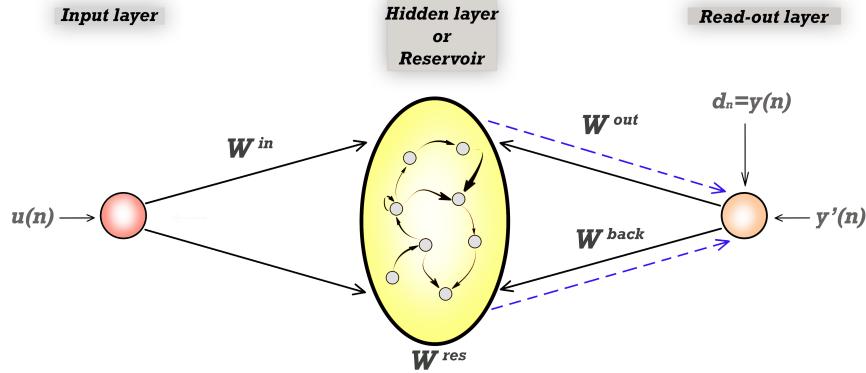


Figure 2.6: **Architecture of classical ESN:** The reservoir is the recurrent neural network with N_x neurons and randomly initialized connections. The reservoir is provided with an input sequence $u(n)$ from the input layer and the teacher signal $y(n)$ from the output layer during training. The input-to-reservoir weights (\mathbf{W}^{in}), output-to-reservoir weights (\mathbf{W}^{back}), and reservoir-to-reservoir weight (\mathbf{W}^{res}) are randomly initialized and remain static during learning. The output weights from the reservoir to output units (\mathbf{W}^{out}) are the only weights learned by the network during training. Adapted from [33]

to be tuned, even though it was shown that RNN works well enough without full adaptation of weights [33]. The classical ESN mainly contains three layers, an input layer, a hidden layer (also known as the reservoir) and a 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 and output-to-hidden layer are optional and depend on the task.

The weights from input-to-reservoir (\mathbf{W}^{in}) and reservoir-to-reservoir (\mathbf{W}^{res}) are sparsely (i.e. most of the elements in these matrices will be zero) and randomly initialized. The sparsity of \mathbf{W}^{in} and \mathbf{W}^{res} is however not a necessary condition but Jaeger [27] recommended the sparse initialization of these weights often give better performance. The non-zero element in the sparse input weight matrix \mathbf{W}^{in} , and the reservoir weight matrix \mathbf{W}^{res} , are usually generated from a uniform or normal distribution [33]. The output-to-reservoir weights (\mathbf{W}^{back}) are also randomly initialized. \mathbf{W}^{in} , \mathbf{W}^{res} and \mathbf{W}^{back} weights remains untrained during the training whereas the weights from the reservoir-to-output layer (\mathbf{W}^{out}) are the only weights learned during the supervised training [27, 33]. 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 previous reservoir state and the previous input on the future reservoir states should vanish gradually [33, 26].

To ensure the echo state property in ESN, firstly, the reservoir weights matrix \mathbf{W}^{res} and the input weights matrix \mathbf{W}^{in} are often generated sparsely (not necessarily) and randomly from a normal or uniform distribution [33]. 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, 33, 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 [33].

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].

$$W_{new}^{res} = \gamma \frac{W^{res}}{\rho(W^{res})} \quad (2.2.1)$$

where W_{new}^{res} 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 [33].

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 [33, 26]. The bigger reservoir size is however not always the 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 sparsely and randomly initialized as specified in the previous section.

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 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’ where $n = 1, 2\dots, T$ is the discrete time step for input sequence of length T . The reservoir then generates a sequence of reservoir states $x(n) \in \mathbb{R}^{N_x}$, 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:

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

$$x(n) = (1 - \alpha)x(n-1) + \alpha x'(n) \quad (2.2.3)$$

where $x(n)$ is the vector of reservoir neuron's activations, $x'(n)$ 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 weights matrix ¹ and $\alpha \in (0, 1]$ is the leaking rate of neurons.

The leaking rate, α , regulates the speed of reservoir state update in discrete time. A smaller value of α induces slow reservoir dynamics thus ensuring the long short-term memory in ESN [33, 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:

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

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:

$$Y' = W^{out}X \quad (2.2.5)$$

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

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 [33]. Usually, a grid search is applied to explore the best parameter combination.

¹this weight matrix is optional and is task specific

Chapter 3

Related Work: θ RARes Model and Its Limitation

Humans have a remarkable ability to perceive and comprehend one or more languages. But, how do they know what does a sequence of symbols mean? 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, θ RARes, to process the sentence across time without having to know about the semantics of the words. The model uses the reservoir computing approach namely Echo State Network (ESN) and has been successfully implemented on iCub humanoid robot platform [21] for the TRA task. It was the first time when ESN was used for the TRA task. The experiments performed using θ RARes model on the TRA task, showed the results toward modeling of language acquisition in brain [23, 20]

3.1 Overview of θ RARes Model

θ RARes Model is basically an ESN to learn and predict the thematic roles of input sentences. The model is based on the notion of grammatical construction [19] which is defined as the mapping of sentence's surface form (word order) to its meaning [20]. Figure 3.1 represents the characterization of thematic role assignment for an active and passive sentence in the grammatical construction. To use the notion of grammatical construction, the model utilize the cue competition hypothesis of Bates et al. [6], which states that the closed class words (e.g. prepositions, articles, determiners, pronouns, etc.), the order of words and the prosody, encodes the grammatical structure of sentences. Thus in order to map the sentence structure to its meaning, the model does not process the raw sentences but, instead use the abstract grammatical form of the sentences [20, 15]. The abstraction to grammatical form marks each word of a sentence into two kinds of symbols: Functional Words (FWs) and Semantic Words (SWs). Functional words are the closed class words or grammatical words. A closed class is the one which contains a limited number of words, and new words are rarely added to it. In Linguistics, a closed class usually includes determiners, prepositions, articles, pronouns and verb

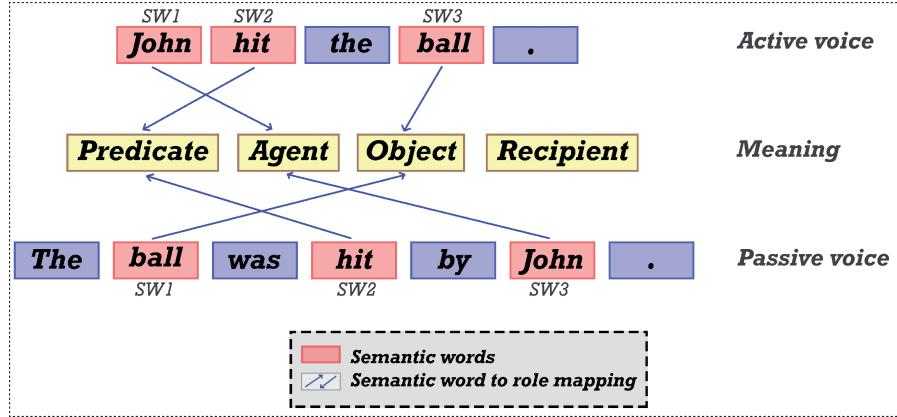


Figure 3.1: Schematic characterization of thematic roles in grammatical construction: The figure shows the mapping (shown with arrows) of semantic words (shown in red) to their thematic roles for an active and passive sentence. The identification of the thematic role of each semantic word in a sentence is defined as the coded meaning of the sentence. Notice that the coded meaning of an active and passive sentence is different, i.e., SW1 is an agent in active sentence whereas the same SW1 is an object in the passive sentence. Despite having the different coded meaning both the sentence share the same sentence meaning, i.e., hit (John, ball). Figure adapted from [20].

inflections like -ed, -ing or -s, etc. [2]. 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].

Figure 3.2 shows the functional organization of the θ RARes model. A set of closed class words is first 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 pushed to a FIFO memory stack (see fig. 3.2) and replaced with a unique token “SW”. The functional words are left unchanged as the functioning of model completely depends on them [20]. This transformation of the sentences to grammatical form by replacing the semantic words with “SW” tokens allows the θ RARes model to be trained on a small dataset. This is because one grammatical form can represent several sentences just by replacing the “SW” token with different nouns, verbs, etc.

To train the θ RARes 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 (see table 3.1). The localist word vectors are input to the ESN, and the coded meaning of the input sentence is teacher-forced on the output layer of ESN (see fig. 3.2). The coded meaning is defined as the description of the thematic roles of each semantic word in a sentence [20]. The model learns the thematic roles of all the semantic words by learning reservoir-to-readout weights (W^{out}) 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

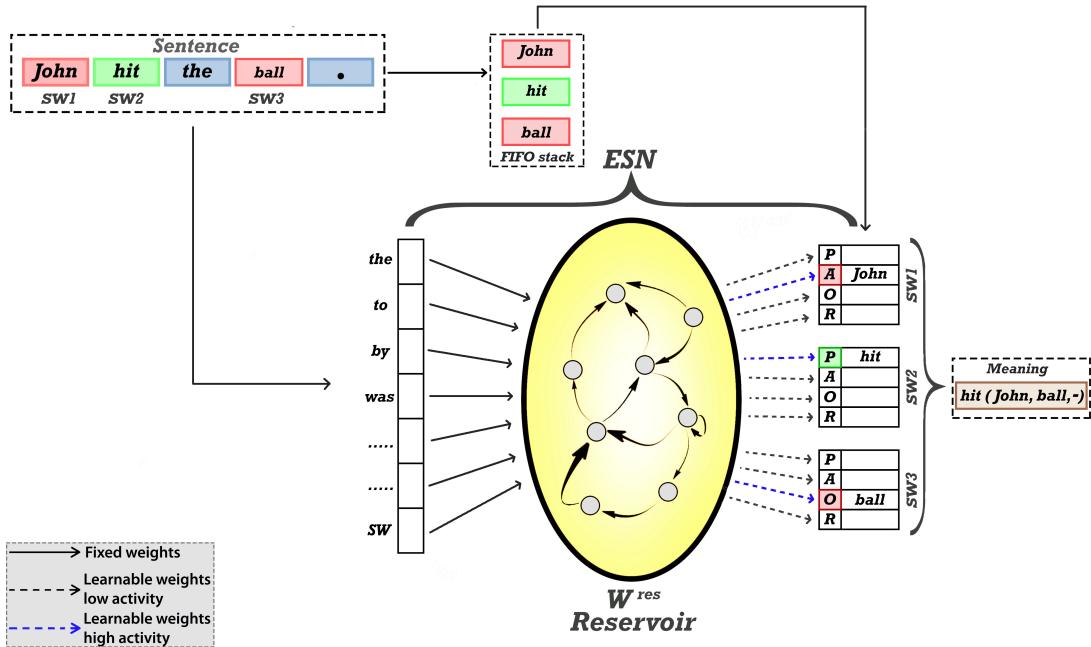


Figure 3.2: Functional organization of θ RARes language model: The figure shows the processing of a grammatical construction by θ RARes model. The semantic words (shown in red and green) are separated at the input and stored in a FIFO memory stack. Closed class words and a “SW” token is input to the model. The model process a sentence, word-by-word, using the localist words vectors as an input. During the training, the coded meaning (i.e. SW1: Agent, SW2: Predicate, SW3: Object) is teacher forced on the output layer, and the readout weights (shown in dotted lines) are learned. While testing the model predicts the coded meaning of the sentence. The meaning of the sentence i.e. hit (John, ball, -) can then be obtained by mapping the coded meaning of a sentence with the semantic words from memory stack. Figure adapted from [20].

Table 3.1: Transformation of a sentence to abstract grammatical form and encoding it using localist word vectors.

| Original words | Grammatical form | Localist vectors | | | |
|----------------|------------------|------------------|----|-----|----|
| | | please | SW | the | on |
| please | please | 1 | 0 | 0 | 0 |
| put | SW | 0 | 1 | 0 | 0 |
| the | the | 0 | 0 | 1 | 0 |
| ball | SW | 0 | 1 | 0 | 0 |
| on | on | 0 | 0 | 0 | 1 |
| the | the | 0 | 0 | 1 | 0 |
| box | SW | 0 | 1 | 0 | 0 |

the activation at last time step. For each “SW” the role that crosses the defined threshold and 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.

3.1.1 Limitations of θ RARes model

Transforming a sentence into its abstract representation and using the localist representation of each word in a sentence (see table 3.1) as an input to the model does not allow the model to leverage the semantic information of the 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.

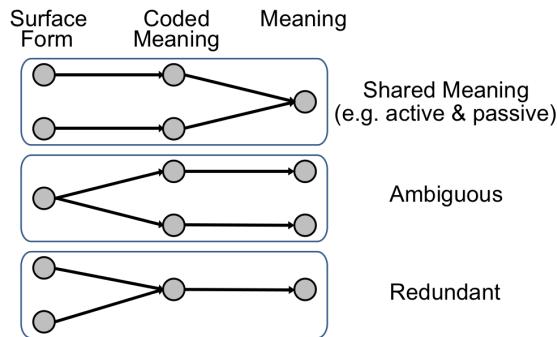


Figure 3.3: Relation between surface form and meanings: Sentence structure to meaning mapping exist in three categories. An active and passive sentence like “John chased the dog” and “The dog was chased by John” have the different surface form but share the same meaning, i.e., chased(John, dog). Also, the coded meaning of both the sentences are different, i.e., SW3 (“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 the redundant sentences have the different surface form but the coded meaning and sentence meaning. Figure adapted from [20].

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)

All the three sentences have 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 θ RARes model with sentence 1: In 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 (semantic 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 | | | | SW-2 | | | | SW-3 | | | |
|-------------|---|---|---|------|---|-------------|---|------|------------|---|---|
| A | O | C | I | A | O | C | I | A | O | C | I |
| <i>take</i> | | | | | | <i>blue</i> | | | <i>box</i> | | |

Training θ RARes model with sentence 2: In the sentence 2, we have three semantic words; SW-1: take, SW-2: left, SW-3: ball. Notice that only the second semantic word (SW-2) in both the sentences (1 and 2) different. Now suppose we train this sentence with a teacher output as follows where the symbols ‘A’, ‘O’, ‘C’ and ‘I’ have the same meaning as described above.

| SW-1 | | | | SW-2 | | | | SW-3 | | | |
|-------------|---|---|---|------|---|---|-------------|------|------------|---|---|
| A | O | C | I | A | O | C | I | A | O | C | I |
| <i>take</i> | | | | | | | <i>left</i> | | <i>box</i> | | |

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 word is treated as a discrete atomic symbol, which does not carry any semantic information about the input word [24].

Testing θ RARes model with sentence 3: Now, when sentence 3 is used for testing, the network may suggest 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. 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 | $C(0.5)$ | $I(0.5)$ |
| | | <i>green</i> | <i>green</i> |

Similarly, the following three sentences also have the same surface 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 the 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 θ RARes model on sentence 1: When using the first sentence for training, the model learns that the fourth semantic word, *London*, is a ‘*Location*’.

Testing θ RARes 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 θ RARes model. Replacing the semantic words with “SW” token and treating each word in the sentence as a unique identifier, removes the semantic and syntactic information carried out by a word in a language [24]. 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 a ‘*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 a 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) [34, 35].

3.2 Research Hypothesis

Transformation of a sentence to its abstract grammatical form and using the localist word vectors as an input to the θ RARes 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. Therefore, inspired by the limitations of θ RARes model described above, this research work hypothesizes that using the distributed word embeddings (e.g. Word2Vec word embeddings) could improve the performance of the model for the TRA task. The rationale behind using distributed word embeddings to represent the words is the fact that they capture and encode semantic and syntactic information carried out by words [34, 51]. Another advantage of using distributed word vectors is that the semantically related words remain close in the neighborhood within a word embedding space (see fig. 2.3) [34]. This clustering of semantically related words is however not possible when the words are represented using localist word vectors. Therefore, the use of distributed word embeddings could allow the θ RARes 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. 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 [35]. 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, the roles could be assigned to the words more accurately although the sentences have the same surface form.

Chapter 4

Approaching Word2Vec- θ RARes Language Model

So far we have seen the basics of Word2Vec model and the Echo State Network. We have also seen the functioning and possible limitations of the θ RARes model. In this chapter, we will see the models proposed to validate the research hypothesis formulated in the previous chapter. First the Word2Vec- θ RARes neural language model in an extension to the θ RARes model will be described, and then the Word2Vec-ESN classifier that could also be used for the TRA task will be discussed.

4.1 Word2Vec- θ RARes Language Model

This research proposes a Word2Vec- θ RARes language model for the TRA task. The proposed model is inspired from the θ RARes model proposed by Hinaut et al. [20] for TRA task. Figure 4.1 shows the basic architecture of Word2Vec- θ RARes model. The Word2Vec- θ RARes model is a 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 are then input to ESN which further processes these word embeddings to learn and predict the thematic roles of all semantic words in the input sentences.

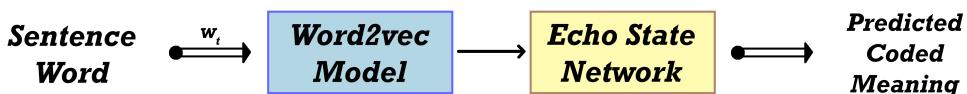


Figure 4.1: **Architecture of Word2Vec- θ RARes model:** The model takes the words of a sentence as an input across time. Word2Vec model generates the distributed vector representation of the input words. ESN then uses the generated word vectors for further processing to learn and predict the coded meaning of an input sentence.

Model initialization: Before using the Word2Vec- θ RARes model for the TRA task, the Word2Vec model is trained using the skip-gram negative sampling [34] 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.

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 the training [25, 33]. 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 randomly to F_{hh} other reservoir neurons and each input neuron is connected randomly to only F_{ih} reservoir neurons.

4.1.1 Training Word2Vec- θ RARes model

The Word2Vec- θ RARes 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 were used to evaluate θ RARes model [20]. Thus it enables us to compare the performance of both Word2Vec- θ RARes and θ RARes model.

Figure 4.2 shows the neural comprehension of the Word2Vec- θ RARes model for thematic role assignment. Like θ RARes model, a set of closed class words is created prior to training of the 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 a 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 coded meaning of input sentence.

The size of the readout layer is $N_{sw} \times N_{tr} \times N_{verb}$: N_{sw} and N_{verb} are respectively the maximum number of semantic words and verbs any sentence can have in the corpus and N_{tr} is the number of possible thematic role assignment for a semantic word. Thus each output neuron encodes the thematic role of a semantic word. In the figure 4.2, $N_{tr} = 4$ where the thematic roles Predicate, Agent, Object and Recipient are represented as P, A, O, R respectively. The model could also take the topologically modified but equivalent coded meaning of the sentences (see fig. 4.3), 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 layer contains $N_{noun} \times N_{verb}$ neurons where N_{noun} is the maximum number of nouns any sentence can have in the corpus.

The Word2Vec model receives the words from an input sentence over time and generates a word vector of E_v dimension which is then used as an input to the ESN. The input layer of the ESN uses this word vector as input features for learning

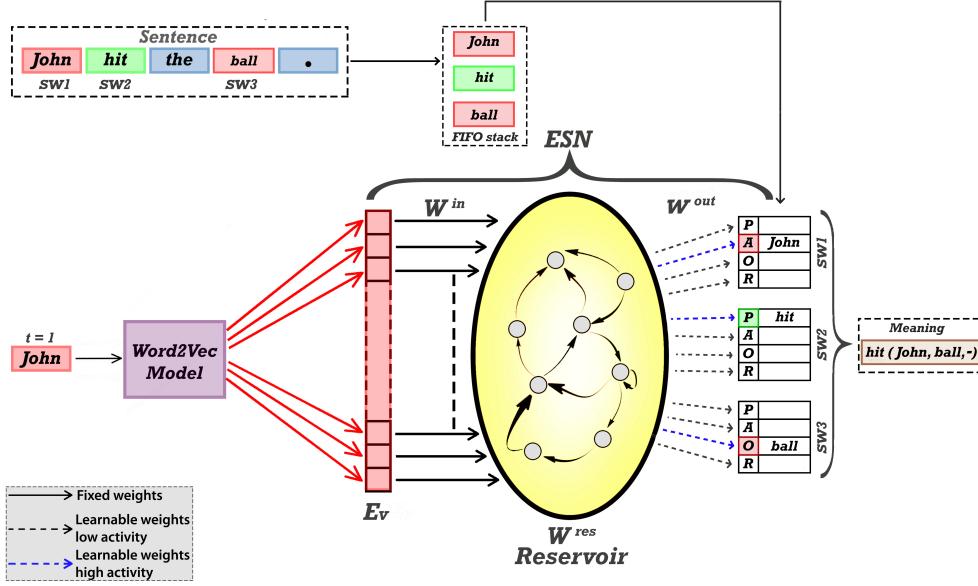


Figure 4.2: **Word2Vec-θRARes language model:** The figure shows the processing of a sentence at time step 1. Semantic words (specified in red and green respectively) are stored in a memory stack at the input. The word ‘John’ is input to Word2Vec model which generate a word vector of E_v dimensions. 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 (SW1: Agent, SW2: Predicate, SW3: Object) and readout weights (W^{out}) are learned. During testing, the readout units predict the 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]

and predicting the thematic roles of the semantic words in the sentences. Thus the size of the input layer is same as the dimension of word vector i.e. E_v . During the training of the model, the readout layer of ESN is also teacher-forced with the coded meaning of the input sentence. The output neurons have an activation of 1 if the corresponding thematic roles are present for the sentence, -1 otherwise. The output of the reservoir is accumulated for each time step during the presentation of a sentence. The accumulated reservoir states are then linearly combined with the readout activations (coded meaning of the input sentence) to learn the reservoir to readout (W^{out}) weights using the linear or ridge regression. The reservoir states of the ESN are reset before the presentation of the consequent sentences. The 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 in the sentence. Thus, the regression is applied from the onset

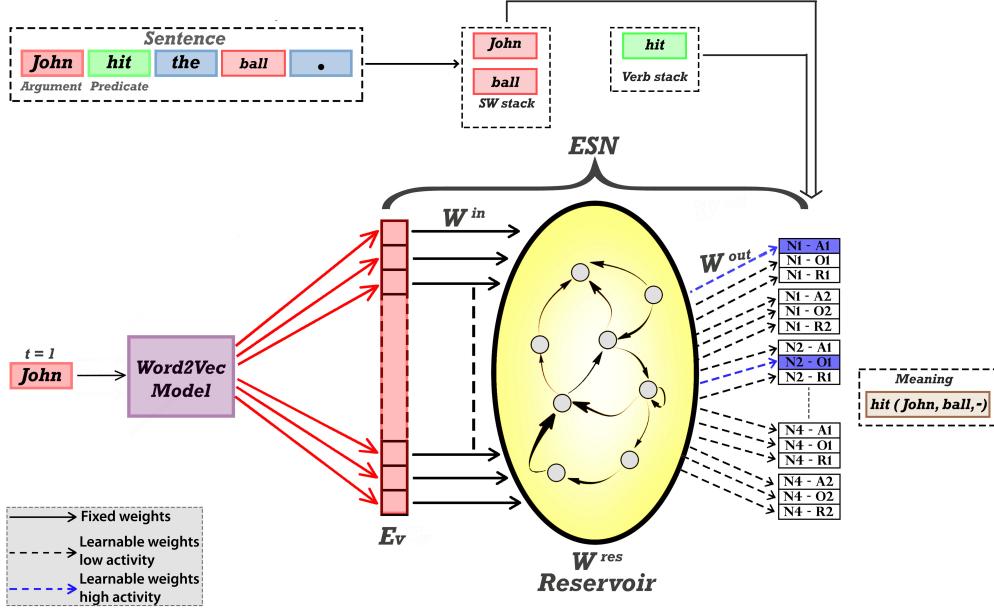


Figure 4.3: Word2Vec-θRARes Language model with topologically modified coded meaning: The figure shows the processing of a sentence with the topologically modified coded meaning at time step 1. Nouns and verbs (specified in red and green respectively) are stored in separate memory stack at the input. 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 input sentence (N1-A1, N2-O1) and readout weights (W^{out}) are learned. The output unit ‘N1-A1’ represents that the first noun is an agent of the first verb in a sentence. During testing, the readout units predict the coded meaning of a semantic word with respect to a verb in an input sentence. The meaning “hit(John, ball, -)” is decoded from coded meaning by mapping the thematic roles with semantic words from memory stacks. Adapted from [20]

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.

4.1.2 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. The readout neurons of the model encode the thematic roles for all semantic words. Thus the output activations produced by the model while testing a sentence are thresholded at 0 for every semantic word. The maximum of all the activations between N_{tr} possible roles 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 actual 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 each semantic words forms the coded meaning of the sentence. The coded meaning of the sentence can then be mapped to the semantic words in FIFO memory stack to obtain the actual meaning of the sentence.

4.1.3 Evaluation metrics

To evaluate the performance of the Word2Vec- θ RARes language model, the same metrics that was used by Hinaut et al. [20] to evaluate the performance of the θ RARes model was employed. The metrics compute the Meaning Error (ME) and the Sentence Error (SE). ME is the percentage of semantic words with incorrect coded meaning, whereas the SE 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]. SE is a stricter measure than ME 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 the Word2Vec- θ RARes model, not all the read-out neurons are considered. Also, recall that if there are more than one action in a sentence, then each semantic word can have a role with respect to each action. Therefore, if a sentence has only three semantic words and two actions, then only the readout neurons corresponding to these three semantic words with respect to both the action are considered to evaluate the performance of the model. Moreover, the coded meaning of remaining neurons is ignored [20].

4.2 Word2Vec-ESN Classifier

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

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

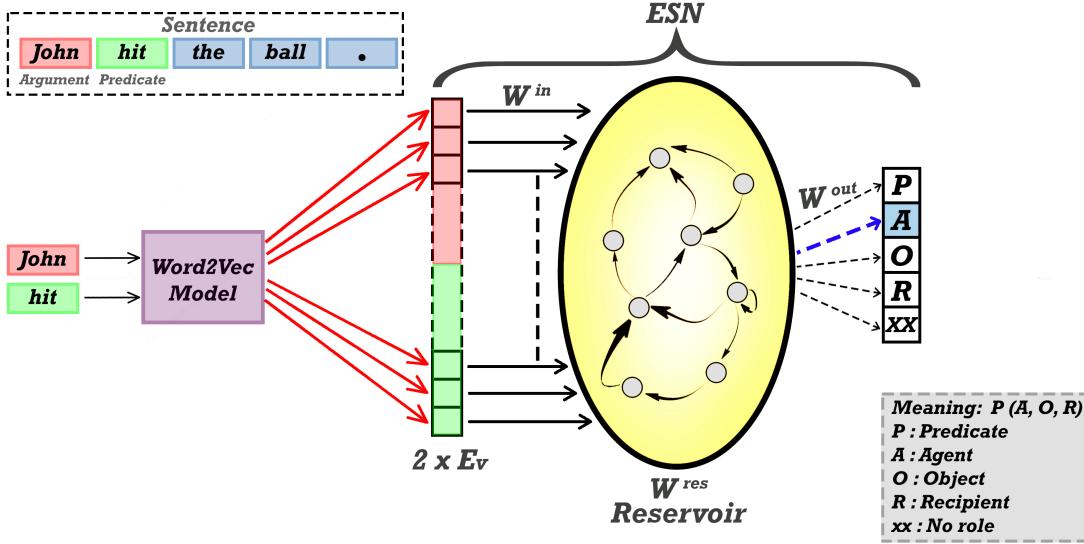


Figure 4.4: Functional organization of Word2Vec-ESN classifier for the TRA task: At any instant of time, an argument-predicate pair (marked in red and green respectively) is input to the model. Word2Vec model generates a vector of $2 \times E_v$ dimensions. ESN takes the resultant vector for further processing. During the training, the readout neurons are presented with the role of input argument-predicate pair (e.g. ‘A’ for Agent) and the readout weights (W^{out}) are learned. While testing, the readout unit predicts the role of argument-predicate pairs, which are then accumulated and decoded at the end of input sentence to form the meaning $hit(John,ball,-)$. Inspired from [56]

predicate describes the verb with respect to which an argument is processed [56]. So, if there are N_v verbs in a sentence, then the same sentence is processed N_v times. Thus each argument-predicate pair takes a role in a sentence. For example in the following sentence there are two predicates namely ‘chased’ and ‘ate’. Therefore, this sentence will be processed twice, and each argument will take a role with respect to a predicate [56].

| | | | | | | | | | |
|---------------------|-----|-----|------|--------|-----|-----|-----|-----|-----|
| 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 a syntactic parser or Part Of Speech (POS) tagger. This dependency on the parser or tagger is a drawback of this model which limits the classifier from being an end-to-end system for the TRA task.

4.2.1 Training Word2Vec-ESN classifier

To train the Word2Vec-ESN classifier, training sentences are presented to the model sequentially. An input sentence is processed as many time as there are verbs in a sentence, forming multiple sequences. Figure 4.4 shows the processing of an example sentence by Word2Vec-ESN classifier. The classifier takes an argument-predicate pair across the time as an input. 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 $2 \times E_v$: the first E_v neurons takes the vector representation of the argument and remaining E_v neurons for the predicate. The readout layer has five neurons each coding for a role (P, A, O, R, XX). During the training, the role of the input argument-predicate pair is also teacher-forced to the model. Output neurons have an activation 1 if an input argument-predicate pair has the corresponding role, -1 otherwise. 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 processes 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 given argument-predicate pair. The role having the highest output activation is considered as the role of argument-predicate pair [3]. The roles for all the argument-predicate pairs in a sentence are collected. 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 is used to deduce the meaning of the sentence as “hit(John, ball, -)”.

4.2.3 Evaluation metrics

To analyze the performance of Word2Vec-ESN classifier on the TRA task, confusion matrix or contingency table [1] is used. Using the confusion matrix, the classification scores: Accuracy, Precision, Recall, and F1-Score, are calculated for all the 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 [38]. 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 the Word2Vec-ESN classifier on the 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-O} | TP_R | E_{R-P} | E_{R-XX} |
| | P | E_{P-A} | E_{P-O} | 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 to the predictions made by the model. The top-left to bottom-right diagonal elements of this matrix represents 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 [48].

$$Accuracy = \frac{\text{number of words correctly labelled}}{\text{total number of words}} \quad (4.2.1)$$

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 the models which merely predict the most frequent class [50, 54]. Hence, the accuracy measure cannot be used alone to evaluate the performance of the model. Therefore, 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 accuracy of a role provided that a specific role has been predicted [48].

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (4.2.2)$$

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*'. [48].

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (4.2.3)$$

Recall for the role ‘A’, from the above confusion matrix is calculated 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 precision and recall. The F1-score measure takes the false positive and the false negative into consideration. This score is useful whenever there is a class imbalance in the dataset [48] and is calculated as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4.2.4)$$

Chapter 5

Material and Methods

This chapter presents the corpora used in this research work and the necessary configuration required to use the Word2Vec- θ RARes model and the Word2Vec-ESN classifier for the TRA task. The chapter is organized as follows. First, we will have an overview of the corpora used to carry out the experiments. Then we will see the initialization of the proposed models, necessary before using the models for the TRA task.

5.1 Corpora and Pre-processing

This section describes the corpora used to perform the experiments with the proposed models for the TRA task. The section also describes the corpus used to train the Word2Vec model.

5.1.1 Corpora for the TRA task

In order to have a fair comparison between the Word2Vec- θ RARes model and θ RARes model on the TRA task, the same corpora ¹ used by Hinaut et al. [20, 21] with θ RARes model was utilized. Thus, the corpus-45, corpus-373, corpus-462, corpus-90582 containing 45, 373, 462 and 90582 sentences respectively, were used to perform the experiments with the proposed models.

Corpus-45: Corpus-45 is a small corpus of 45 sentences. This corpus contains the sentences in grammatical forms, i.e., ‘N’ and ‘V’ tokens were used respectively to represent the nouns and the verbs [20]. The corpus also includes the coded meaning of each sentence. The corpus contains the sentences 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
2. the N was V by the N. # N1-O1 N2-A1

¹<https://sites.google.com/site/xavierhinaut/downloads>

The first part of the construction (before ‘#’ token) is the sentence in its grammatical form. The coded meaning of the sentences is represented after the ‘#’ token. The coded meaning ‘N1-A1’ can be interpreted as; the first noun is an agent of the first verb. The symbols ‘O’ and ‘R’ were used respectively for the roles object and recipient.

Corpus-462 and Corpus-90582: The sentences in the corpus-462 and corpus-90582 were generated by Hinaut et al. [20] using a context-free grammar for English language and used for the TRA task. 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 can be 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 corpora is divided into four parts. The first part (before ‘<*o*>’) 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 ‘#’) contain the raw sentence with verb inflections (i.e. ‘-s’), and the fourth part is the abstract representation of a sentence with the semantic words removed and replaced with the 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 and sentence meaning (see fig. 3.3). In total, there are only 2043 distinct coded meanings [20]. This corpus also has an additional property that along with the complete coded meanings of the sentences, it also include sentences with incomplete meanings. For example, the sentence “The Ball was given to the Man” have no ‘Agent’ in it, 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 gathered 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 to perform the same actions again in the natural language. Thus the corpus contains 373 simple or complex instructions to perform single or double actions with temporal correlation (see Action Performing task in Hinaut et al. [21] for more details). For example, the instruction “Point to the guitar” is 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 the semantic words from the sentences is also provided with this corpus [21]. Also, unlike corpus-462 and corpus-90582 this corpus does not contain verb inflections. Thus, this corpus is complex enough to test the learnability and generalization ability of the proposed models on the TRA task.

Pre-processing: As described earlier, unlike the θ RARes 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”). Therefore, these sentences in the corpus-462 and corpus-90582 were processed to obtain the raw sentences without verb inflections. Firstly, all the words were lowercased and then the verbs with inflections were 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 were imposed previously using verb inflections e.g., $\text{vector}(\text{'walks'}) - \text{vector}(\text{'walk'}) \approx \text{vector}(\text{'talks'}) - \text{vector}(\text{'talk'})$.

5.1.2 Corpus for training Word2Vec model

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 the words. The corpus contains 2,65,8629 unique words. The reason for choosing the Wikipedia dataset to train the model is that it contains most of the words used in the English language. Thus we can have the vector representation of almost all

²service used to obtain verb conjugations: <http://www.scientificpsychic.com/cgi-bin/verbconj2.pl>

³<https://dumps.wikimedia.org/enwiki/latest/>

the words in the English language. Also, the Word2Vec model does not give good quality word vectors when trained over a small corpus. Thus a general purpose data set with billions of words is required to have better word embeddings. Once the vector representation of the words in the Wikipedia dataset is obtained, they can also be updated by training the model further on the domain specific dataset (corpus-462 and corpus-90582, etc.) with more bias towards the domain specific dataset.

5.2 Experimental Setup

5.2.1 Obtaining distributed word embeddings

Before using the proposed models for the TRA task, the distributed embeddings of the words are required to be learned by the Word2Vec model. Thus the Word2Vec model is first trained on the Wikipedia dataset using skip-gram negative sampling (see Chapter 2 for more details). Skip-gram negative sampling was used because it was claimed to accelerate the training of the model and the resultant word vectors perform better on the word analogy task [35, 34].

To obtain the distributed 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 learns the distributed word vectors corresponding to the size of the hidden layer. All the six models were trained using the same hyperparameters as follows. A standard context window of ± 5 was used. A bigger context window could also be used to improve the quality of word vectors but at the cost computation time [34]. The negative sampling size of 5 was chosen i.e. five 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 before training the model. To update the network weights, the stochastic gradient descent was used [45, 34] with the initial learning rate of 0.025, which drops to 0.0001 linearly as training progress. These parameter were chosen by following the heuristics discussed in [35, 34].

The word embeddings obtained by training the Word2Vec model on the Wikipedia dataset are accurate enough to capture the semantic relationship of the words for e.g. $\text{vec}(\text{'Paris'}) - \text{vec}(\text{'France'}) + \text{vec}(\text{'Germany'}) \approx \text{vec}(\text{'Berlin'})$. Before training the model on the 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 train the Word2Vec model further, 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. To facilitate online training of the Word2Vec model, there is a need to update the vocabulary of the model

if some words are not already present in the vocabulary. Unfortunately, neither the C++ API ⁴ nor the Gensim python API [44] implementation of Word2Vec model supports the vocabulary update, once created. So the online training ⁵ of Word2Vec model was implemented 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 will have poor quality if their count in the new corpus is less. Thus, to have more impact of the domain specific corpus on the word 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-373, corpus-462 and corpus-90582). While training 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 are normalized using L-2 norm before using them for TRA task. One important point to remember while training the Word2Vec model is that if an already trained model has to be trained further on any other corpus, then the word vectors should not be normalized, as it is not possible to train the model again with normalized word vectors [44]. 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 [33]. Thus for the proposed models, a reservoir of 1000 leaky integrator neurons with \tanh activation function was used (until and unless specified). 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, the optional output-to-reservoir weights (W^{back}) were not used. Therefore, the reservoir state update equations 2.2.2 and 2.2.3 thus changes to:

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

$$x(n) = (1 - \alpha)x(n-1) + \alpha x'(n) \quad (5.2.2)$$

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

⁴<https://code.google.com/archive/p/word2vec/>

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

⁶Idea discussed on: <https://groups.google.com/forum/#topic/gensim/Z9fr0B88X0w>

neurons and each input neuron was randomly connected to only 2 other reservoir neurons. Using the fixed fanout scales the computational cost of reservoir state update linearly with increase in the reservoir size [33]. 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 [33]. Whereas other parameters like Spectral Radius (SR), Input Scaling (IS) and Leak Rate (LR or α) are task dependent and are often tuned by multiple trials. Thus to identify these parameters for the TRA task, a grid search over the parameter space was performed using five reservoir instances (i.e., reservoir weights were initialized using five different random generator seeds). 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- θ RARes 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 determine the optimal parameters [33]. As both the proposed models (Word2Vec- θ RARes and Word2Vec-ESN classifier) process the sentences differently and have different training objectives, the ESN parameters for both the models were optimized separately. Additionally, the Word2Vec- θ RARes 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- θ RARes model: To optimize the reservoir parameters of this model, 462 sentences of corpus-462 with the topologically modified coded meaning (see fig. 4.3) were 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- θ RARes 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 ≈ 46 sentences). The model was trained on sentences from 9 subsets and then tested on the 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, the optimal parameters were identified 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- θ RARes 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 the optimal parameters were identified as $SR = 1.3$, $IS = 1.0$, $LR = 0.4$.

5.2.4 Input and Output coding

As mentioned in Chapter 4, the Word2Vec- θ RARes model and Word2Vec-ESN classifier process the sentences differently. Therefore, the input and output coding for both the model also differs. However, the initialization of Word2Vec model and ESN reservoir weights remains the same for both the models.

Word2Vec- θ RARes model: A raw sentence is presented to the model, where each word in the sentence is processed, word-by-word across time, by both Word2Vec model and ESN. The Word2Vec model outputs the word vector of $E_v = 50$ dimension, 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 Chapter 6), 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 times 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 of ESN contains 100 neurons where first 50 neurons encode 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- θ RARes, 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 neurons of ESN have an activation 1 if the input word-verb (argument-predicate) pair have the corresponding role, -1 otherwise.

Chapter 6

Experiments and Results

This chapter summarizes the experiments performed with the proposed model described in chapter 4. In total, seven experiments were performed using the corpora outlined in the previous chapter. Along with the experiments, the results are also presented and compared with that of obtained using θ RARes model wherever required.

6.1 Experiment-1: Performance on Small Corpora

In order to see the performance of Word2Vec- θ RARes model and Word2Vec-ESN classifier with the small corpus on the TRA task, the models were first tested with a limited number of sentences i.e. 26 sentences (sentence 15 to 40) from corpus-45. The chosen sentences have the distinct surface form and grammatical structure (e.g. active, passive, dative-passive). Using five instances of both the models, each model was 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 26 constructions used in the experiment by exploring the parameter space and using LoO cross-validation approach. This experiment remains a toy demonstration, and the generalization capability of the model will be investigated later with an extended corpus in Experiment-2 and Experiment-5.

Word2Vec- θ RARes Model: For simulation with Word2Vec- θ RARes model in this experiment, the model was operated in the SCL mode. Also, the reservoir of 1000 neurons and parameters $SR = 2.1$, $IS = 6.6$, $LR = 0.5$ were used. When trained and tested on all the 26 sentences, the model learned the sentences without any errors. However, during the cross-validation the model failed to generalize and yielded high cross-validation error rates ($ME = 35.09\%$ and $SE = 85.38\%$). This indicates that the Word2Vec- θ RARes model is overfitting on the dataset.

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

| | | Word2Vec- θ RARes | θ RARes |
|-----------------------|------|--------------------------|----------------|
| 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-validation errors with both Word2Vec- θ RARes model and θ RARes model (shown in %). Mean and std. is the average and standard deviation of errors over several model instances.

However, this is not surprising because the dataset contains limited examples, which constrained the model to generalize well.

As illustrated in Table 6.1, it can be seen that the cross-validation errors with Word2Vec- θ RARes model are almost doubled when compared to θ RARes model. It is also important to know that the cross-validation errors of θ RARes model reported in the Table 6.1 are averaged over 100 reservoir instances of 100 units each [20]. Whereas, the cross-validation errors of Word2Vec- θ RARes are averaged over five reservoir instances of 1000 neurons each. With the small reservoir of 100 neurons, the Word2Vec- θ RARes failed to learn and generalize (not reported here) on the same 26 sentences used in this experiment. This shows that the θ RARes model performs better than the proposed Word2Vec- θ RARes model on the small dataset even with a small reservoir.

Word2Vec-ESN classifier: For simulation with Word2Vec-ESN classifier, a reservoir of size 500 neurons and parameters $SR = 0.2$, $IS = 0.7$, $LR = 0.7$, were used. As illustrated in Table 6.2, the Word2Vec-ESN classifier learned the 26 sentences resulting in F1-Score of 95.68% during the training. While testing the classifier using LoO cross-validation method it generalized with low F1-Score of 68.53%. The high difference in F1-Score during training and testing conditions points toward the limited generalization capability of the classifier on unseen sentences.

6.2 Experiment-2: Generalization Capabilities

In the previous experiment, we saw the performance of both the proposed models with a limited number of sentences. The results indicated that the model is probably overfitting and not generalizing on the unseen sentences. Therefore, in this experiment, an extended corpus of 462 sentences (see section 5.1) was used to test the generalization capability of the models. The simulations were performed with

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

| | | Precision | Recall | F1-Score |
|-----------------------|------|-----------|--------|----------|
| training error | mean | 94.92 | 96.54 | 95.68 |
| | std. | 00.27 | 00.74 | 00.46 |
| test error | 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.

five instances of both the proposed models each with a reservoir of 1000 neurons. To test the learning ability, both the models were initially trained and tested on all the sentences of the corpus-462. Also, to test the generalization ability of both the models, 10-fold cross-validation was used.

Word2Vec- θ RARes model: In order to compare the learnability and generalization capability of the Word2Vec- θ RARes and θ RARes model, simulations were performed in both the SCL and SFL mode. Also, the reservoir parameters identified during grid search for both the learning modes were used i.e. $SR = 2.4$, $IS = 2.5$, $LR = 0.07$ in SCL mode and $SR = 2.2$, $IS = 2.3$, $LR = 0.13$ SFL mode (see section 5.2.3 in chapter 5).

Table 6.3 illustrates the train and test errors obtained with Word2Vec- θ RARes and θ RARes model. One can see that while training both the Word2Vec- θ RARes and θ RARes model learned all the sentences thoroughly (less than 1% error rates) in both the SCL and SFL mode. Comparing the performance during testing, we can see an improvement of 8.04% sentence error in SCL mode with Word2Vec- θ RARes 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- θ RARes and θ RARes 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- θ RARes model performs better in SCL mode when compared to SFL mode, whereas it is vice-versa in θ RARes model. Additionally, the Word2Vec- θ RARes model outperform the θ RARes model in SCL mode.

The same experiment was also performed using the sentence in corpus-462 without the topologically modified coded meaning. The cross-validation errors were found to be negligibly increased ($\approx 1\%$ in SCL mode and $\approx 3\%$ in SFL mode) but with the different reservoir parameters ($SR = 3.9$, $IS = 3.8$, $LR = 0.1$ in SCL mode and $SR=3.2$, $IS=3.9$, $LR=0.1$ in SFL mode). This supports the suggestions of Hinaut et al. [20], that topologically modified coded meaning of a sentence (see fig. 4.3) is equivalent to the usual coded meaning (see fig. 4.2). Even without

Table 6.3: Meaning and Sentence error in the SCL and SFL modes.

| | | Word2Vec- θ RARes | | | | θ RARes | | | |
|------------------|------|--------------------------|-------|---------------|-------|----------------|-------|---------------|-------|
| | | 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- θ RARes and θ RARes model [20]. The errors are given in percentage upto two decimal precision. SCL: Sentence Continuous Learning; SFL: Sentence Final Learning; std: standard deviations. Simulations were done with 5 model instances each with a reservoir of 1000 neurons.

the modified coded meaning the Word2Vec- θ RARes model performed better than θ RARes model in SCL mode.

Word2Vec-ESN classifier: Recall that unlike Word2Vec- θ RARes model, the performance of Word2Vec-ESN classifier is evaluated using classification metrics (see section 4.2). The classifier process the raw sentences and the distributed vector representation of the input argument-predicate pair is fed into ESN. In order to see the effect of distributed word vectors over localist word representation, it is also important to test the behavior of the classifier on the sentences transformed to grammatical form and using localist word representation as an input to ESN. Thus for this experiment, simulations were performed with the Word2Vec-ESN classifier in two configurations mentioned below.

- Configuration-1:** In this configuration, the Word2Vec-ESN classifier process the raw sentences and distributed vector representation of the 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.
- Configuration-2:** In this configuration, the Word2Vec-ESN classifier was used without the Word2Vec unit, and only the ESN is used for processing the sentences. The input sentences were transformed in the grammatical

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

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

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

form: all the semantic words in the sentence were replaced with ‘SW’ token, and the 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.

Table 6.4 illustrates the training and cross-validation classification scores of simulations performed with the Word2Vec-ESN classifier in both the configurations. The classifier in configuration-1 learned and generalized on the corpus with high classification scores whereas, in configuration-2, the classifier learned the corpus with low classification scores and hence failed to generalize as shown by low scores. Notice the marginal difference between the training and cross-validation scores in configuration-1. This indicates that the Word2Vec-ESN classifier is generalizing with high classification scores on the unseen sentences and do not overfit. Comparing the performance of the classifier in both the configurations, it can be observed that the classifier performed better when the distributed word embeddings were used as an input to ESN (i.e. configuration-1) as compared to the use of localist words vectors as an input to the ESN (i.e. configuration-2).

Figure 6.1 shows the confusion matrices of true and predicted roles for input argument-predicate pairs in configuration-1 and configuration-2. The matrices are plotted using the cross-validation results of the Word2Vec-ESN classifier. The corresponding classification scores of the individual roles produced by the classifier during 10-fold cross-validation are also reported in Table 6.5 for both the configurations.

As shown in the confusion matrix (see fig. 6.1), it can be observed that when using the distributed vectors of argument-predicate pair as an input to ESN (i.e.

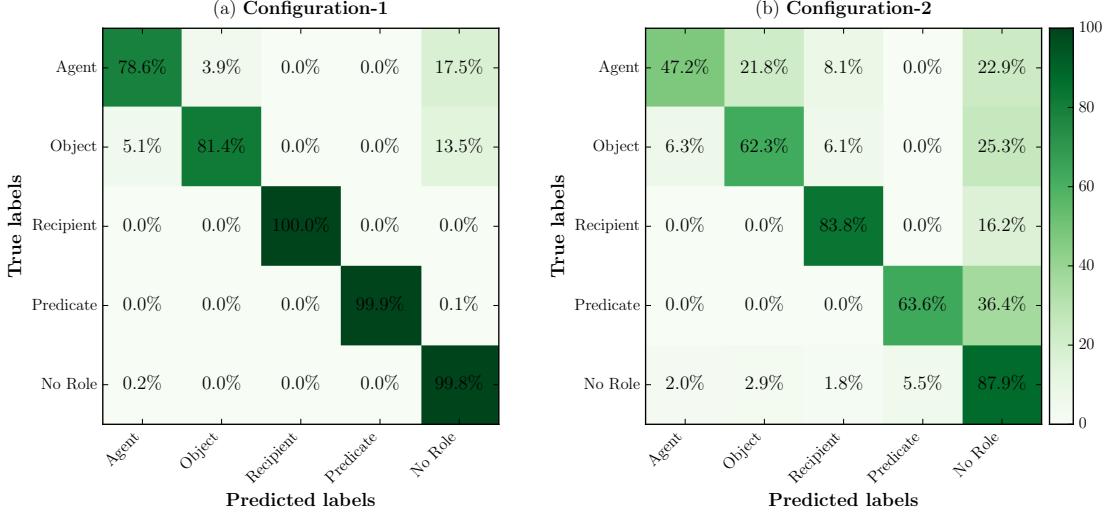


Figure 6.1: Normalized confusion matrix with Word2Vec-ESN classifier in configuration-1 and configuration-2: The confusion matrix with true roles (in rows) and predicted roles (in columns). The top-left to bottom-right diagonal shows the percentage of words whose roles are predicted correctly. Everything 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.

configuration-1), the model predicted the roles ‘Recipient’, ‘Predicate’ and ‘No Roles’ almost without making any errors. Notice that the classifier mispredicted the roles ‘Agent’ and ‘Object’ mostly as ‘No Role’. It can also be noticed that the roles ‘Agent’ and ‘Object’ were confused with each other i.e. the argument-predicate pairs with the role ‘Agent’ were misclassified as ‘Object’ and vice-versa.

Using the sentences in grammatical form along with the localist word vectors of the argument-predicate pairs (i.e. configuration-2), as an input to ESN, the roles ‘Recipient’ and ‘No Role’ were predicted correctly with least errors of 16.2% and 12.1%. Notice that significant amount of all the roles were misclassified as ‘No Role’, where the ‘Predicate’ being the highest role to be misclassified (36.4%) as ‘No Role’. The classifier in configuration-2 also seems to be confused between the roles ‘Agent’ and ‘Object’. Additionally, the argument-predicate pairs with the true roles ‘Agent’ and ‘Object’ were also confused with the role ‘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 errors in predictions as compared to other roles. In both the configurations, the model was mainly confused between roles ‘Agent’ and ‘Object’, but with the higher number of misclassification in configuration-2. Additionally, in configuration-2, the roles ‘Agent’ and ‘Object’ were wrongly predicted as ‘Recipient’, whereas in configuration-1, this misprediction does not exist. Compar-

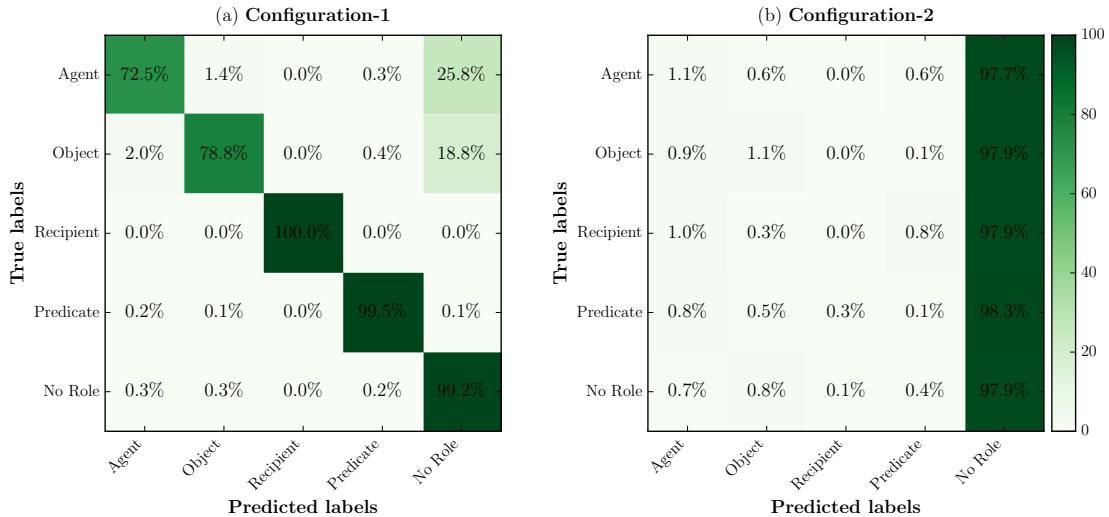


Figure 6.2: Normalized confusion matrix with the Word2Vec-ESN classifier on scrambled corpus-462: The confusion matrix shows the true roles (in rows) and predicted roles (in columns) of argument-predicate pairs. The top-left to bottom-right diagonal shows the percentage of argument-predicate pairs for which the roles are predicted correctly. All cells, other than this diagonal represents the incorrect prediction of the roles. Configuration-1: raw sentences processed by the Word2Vec-ESN classifier and distributed 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.

ing the false predictions made by the classifier in both the configurations, it can be observed that the number of incorrect predictions was comparatively less in configuration-1. Also, in configuration-1, all the argument-predicate pairs with actual roles ‘Recipient’, ‘Predicate’ and ‘No Role’ were predicted almost without any errors whereas, in configuration-2, the classifier misclassified these roles respectively by 16.2%, 36.4%, and 12.1%.

6.3 Experiment-3: Effect of Corpus Structure

As described earlier, 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 are thus possibly utilizing the underlying grammatical structure to some extent for learning and generalization. To test this hypothesis and to demonstrate that the models are not generalizing on any other inconsistent regularity in the corpus, in this experiment, the inherent grammatical structure from the sentences of corpus-462 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.

For the simulations, the corpus-462 with the scrambled sentences (i.e. in the

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

| Role | | Configuration-1 | | | Configuration-2 | | | Support |
|------------------|-------|-----------------|------|------|-----------------|------|------|---------|
| | | Pr | Re | F1 | Pr | Re | F1 | |
| 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.

absence of any grammatical structure) was presented to the Word2Vec- θ RARes model and Word2Vec-ESN classifier. Keeping all conditions and the reservoir parameters same as used in Experiment-2 for both the models, a 10-fold cross-validation was performed with five model instances. Apart from cross-validation, the models were also trained and tested on all the sentences in the corpus to determine the learning ability of the models with the scrambled corpus.

Word2Vec- θ RARes model: As illustrated in Table 6.3, it was observed that when using the scrambled corpus as an input to the Word2Vec- θ RARes model, the model learned the scrambled sentences 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, it generalized with high error rates. While comparing the results of Word2Vec- θ RARes model with and without inherent grammatical structures in the sentences, it can be observed that the model did not perform better on the corpus with the scrambled sentences. This indicates that the model is not generalizing on any other inconsistent regularity in the corpus, but instead utilizing the grammatical structure present in the sentences.

Word2Vec-ESN classifier: Table 6.4 illustrates the effect of the scrambled corpus on the Word2Vec-ESN classifier. It can be observed that in the configuration-1, the classification scores in training and testing conditions, remained almost equivalent to the scores obtained with the unscrambled corpus. Whereas in configuration-2, the Word2Vec-ESN classifier learned and generalized on the scrambled corpus with low classification scores. Also in configuration-2, notice the classification scores in both the training and testing conditions are lower than that obtained with the unscrambled corpus.

Figure 6.2 shows the confusion matrix obtained when the scrambled corpus is exposed 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 role is negligibly affected whereas in configuration-2, the classification scores of individual roles dropped heavily.

6.4 Experiment-4: Effect of Reservoir Size

The most important hyperparameter which affects the performance of the proposed models is the size of the reservoir (i.e. N_x = number of neurons in the reservoir). The addition of neurons in the reservoir is computationally inexpensive because the read-out weights (W^{out}) scales linearly with the number of neurons in the reservoir [52]. Liu et al. [32] argued that as the number of neurons in the neural network increases, so does the danger of overfitting, thus ensuring the failure of the network to generalize during cross-validation. So, in order to determine the effect of reservoir size on the performance of the Word2Vec- θ RARes model and Word2Vec-ESN classifier, the simulations were performed over a range of reservoir sizes ¹ ² with the five instances of the proposed models. The sizes of the reservoir were randomly chosen from a range of 50 to 5500 neurons. Also, for the simulations, corpus-462 was used, and models were tested using 10-fold cross-validation approach.

Word2Vec- θ RARes model: With the Word2Vec- θ RARes model, simulations were performed on a range of reservoir size¹ in both the SCL and SFL modes, using the same optimal reservoir parameters as used in 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.

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 errors continuously

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

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

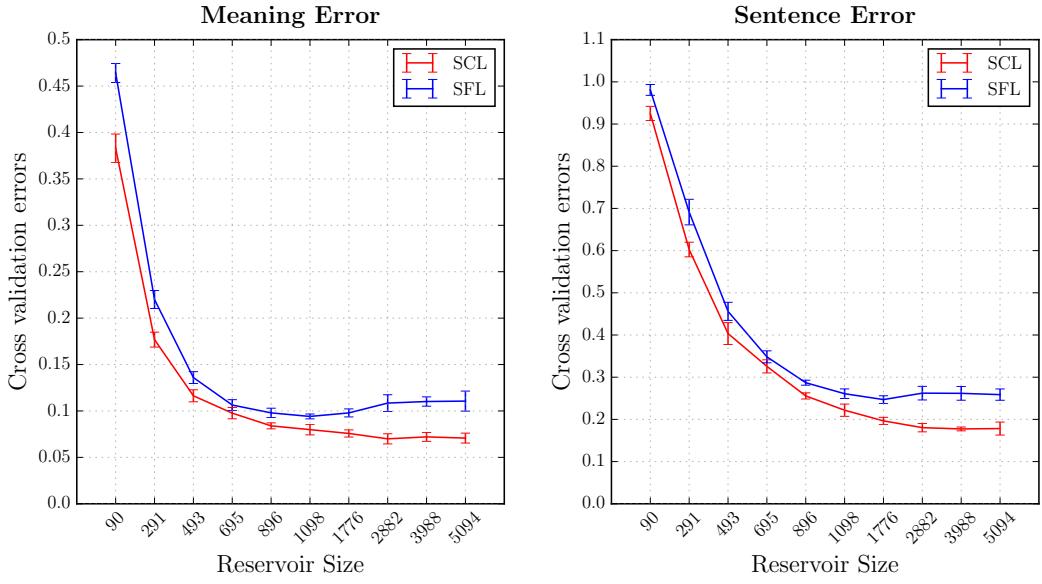


Figure 6.3: Effect of reservoir size on cross-validation errors of Word2Vec- θ RAres model. NOTE: the scale of cross-validation error for the meaning and sentence error are different.

drop 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 and sentence error in both the SCL and SFL mode is negligibly affected. Notice that with the increase in reservoir size, the meaning and sentence error curves follows the similar trend in both the SCL and SFL mode. Another interesting pattern to notice is that, with all the reservoir sizes studied, the meaning and sentence error in SCL mode is always lower than in SFL mode. Overall the important behavior to notice is that both the meaning and sentence cross-validation error rates reduce with the increase in the reservoir size, but asymptotes when the reservoir size is above 1000 neurons.

Word2Vec-ESN classifier: To determine the effect of reservoir size on the Word2Vec-ESN classifier, simulations were performed on a range of reservoir sizes ², in configuration-1 and configuration-2 of the classifier as described in Experiment-2. Also, for the simulations, reservoir parameters were kept identical to that used in Experiment-2.

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 negligible drop of scores. Whereas in configuration-2 i.e. when using the sentences in grammatical form encoded using localist representa-

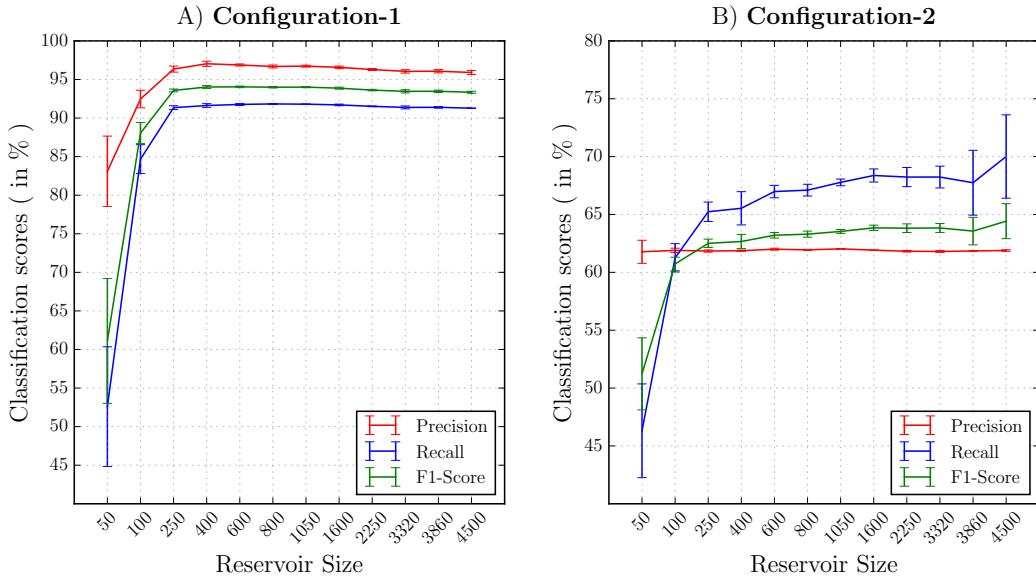


Figure 6.4: Effect of reservoir size on classification scores of Word2Vec-ESN classifier. NOTE: the scale of cross-validation scores in configuration-1 and configuration-2 are different.

tion, the classification scores also improve, with the increase in reservoir size from 50 to 250 neurons. As the reservoir size is further increased from 250 neurons, the recall score is improved whereas the precision remains almost flat with negligible improvement. The F1-Score is harmonic mean of precision and recall, and thus it is also slightly increased. Notice that 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.

6.5 Experiment-5: Scalability of the Model

In order to investigate the effect of corpus size and to 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 [20] which impede the learning and generalization of the model, this experiment will also validate the model's ability to process the ambiguous sentences. Thus, in order to study the scaling capability 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 original corpus of 90582 sentences [20]. Each of the generated sub-corpora was exposed to the Word2Vec- θ RARes 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- θ RARes model in both learning modes i.e. SCL

and SFL, using five model instances each with a reservoir of 5000 neurons. All the other reservoir parameters were kept identical to the Experiment-2.

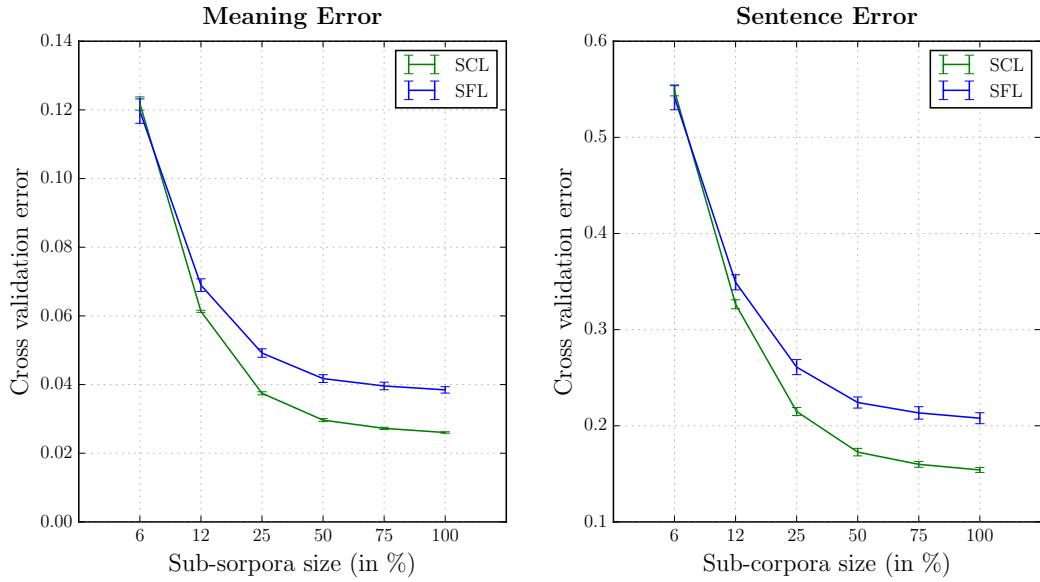


Figure 6.5: Effect of corpus size on cross validation errors of Word2Vec- θ RARes model in SCL and SFL mode. NOTE: the scale of cross-validation error for the meaning and sentence error are different.

Figure 6.5 shows the cross-validation errors rates with respect to corpus size on the Word2Vec- θ RARes model. It can be observed that with the increase in corpus size from 6% to 50%, the meaning error sharply drops from 12.18% to 2.96% in the SCL mode and from 11.96% to 4.17% in the SFL mode. Similarly, the sentence error also decreases from 54.84% to 17.25% in SCL and from 54.14% to 22.41% 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% meaning error and 15.98% sentence error in the SCL mode and with 3.95% meaning error and 21.33% sentence error in the SFL mode. Although the error rates gradually drop when the corpus size is further increased from 25% to 100% but the improvement rate of errors is very less. This indicates that cross-validation errors improve with the increase in corpus size up to a certain limit and any further increase in corpus size barely influence the performance of the model.

Comparing the effect of corpus size on Word2Vec- θ RARes model (see fig. 6.5) and the θ RARes model (see fig. 6.6), it can be observed that in SFL mode, with the increase in corpus size from 6% to 25%, the meaning error in θ RARes model dropped from $\approx 17\%$ to $\approx 3\%$ and the sentence error dropped from $\approx 70\%$ to $\approx 20\%$. Whereas in the Word2Vec- θ RARes 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,

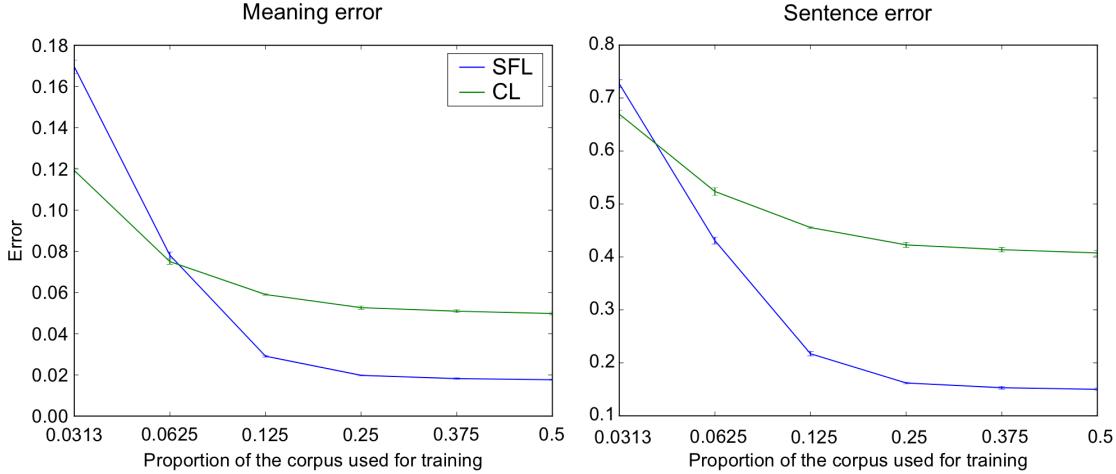


Figure 6.6: Effect of corpus size on cross validation errors using θ RARes model in SCL (legend CL) and SFL learning modes [20]. NOTE: The x-axis represents the corpus-size used for training i.e. if sub-corpora size is 25% then corresponding tick on the axis is 0.125 (12.5%). The scale of cross-validation error for the meaning and sentence error are different.

with the increase in corpus size, the improvement in error rates with θ RARes model in the SFL mode is higher as compared to Word2Vec- θ RARes model.

Comparing the performance of both the θ RARes and Word2Vec- θ RARes model in the SCL mode, it can be seen that the meaning error in latter continuously drops when the corpus size is further increased from 12% whereas in the θ RARes model the decline in meaning error is negligible. The sentence error in θ RARes 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- θ RARes model. From the lower to the upper limit of corpus size studied, the sentence error with the θ RARes model in SCL mode dropped from $\approx 65\%$ to $\approx 40\%$ whereas, in the Word2Vec- θ RARes model it dropped from $\approx 55\%$ to $\approx 15\%$. The meaning error in SCL mode, on the other hand, dropped from $\approx 12\%$ to $\approx 5\%$ in the θ RARes model whereas, in the Word2Vec- θ RARes model it dropped from $\approx 12\%$ to $\approx 2\%$.

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

6.6 Experiment-6: Generalization on New Corpus

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

Word2Vec- θ RARes model: To test the generalization of Word2Vec- θ RARes 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 the results of this experiment can be compared with that of obtained using θ RARes 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 the optimal reservoir parameters, 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 also enable us to test the robustness of these model parameters on the new corpus i.e. corpus-373.

Table 6.6 reports the mean sentence error and best generalization error values obtained with the Word2Vec- θ RARes and θ RARes model. The best generalization error here represents the percentage of sentences whose meanings were predicted incorrectly in common by all 10 model instances [21]. In comparison to the θ RARes model, the mean sentence error in the Word2Vec- θ RARes 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- θ RARes model generalized better with both the reservoir sizes (500 and 1000) as compared to the θ RARes model. Notice the improvement in both the models with the increase in reservoir size from 500 to 1000 neurons. It can be seen that the improvement in cross-validation error in the Word2Vec- θ RARes model is comparatively less than the θ RARes model. With the increase in reservoir size from 500 to 1000, the mean sentence error and the best generalization error in θ RARes improved by 10.7% and 9.65% respectively. Whereas with the Word2Vec- θ RARes model, the mean sentence error and the best generalization error improved by 2.36% and 0.81% respectively. This suggests that the generalization ability of the Word2Vec- θ RARes model on corpus-373 is negligibly affected by the increase in reservoir size.

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 neu-

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

| Reservoir | Error | Word2Vec- θ RARes Model | θ RARes 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 generalization errors (in %) obtained with the Word2Vec- θ RARes model and the θ RARes in SCL mode with reservoir of 500 and 1000 neurons. The results reported are mean and standard deviation of errors obtained with 10 model instances. The best generalization error here means the percentage of sentences predicted incorrectly in common by all 10 reservoir instances [21].

rons (see fig. 6.7), all the ten instances of Word2Vec- θ RARes and θ RARes model correctly predicted the meaning of 146 and 35 sentences respectively. Whereas the meaning of 99 and 166 sentences out of 373 was wrongly predicted by all the ten instances of Word2Vec- θ RARes and θ RARes model.

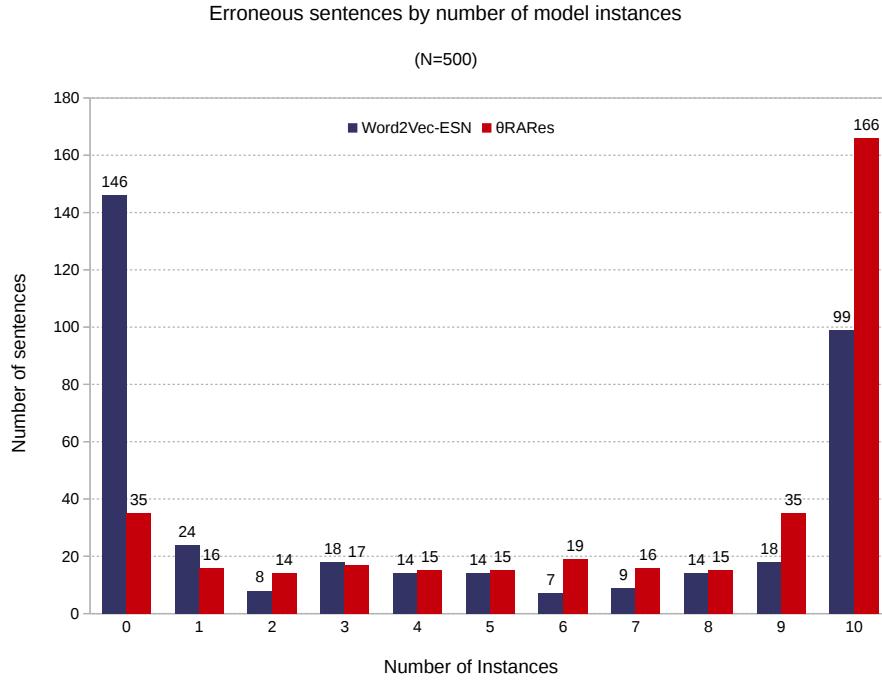


Figure 6.7: Sentence error count by number of instances with reservoir of 500 neurons.

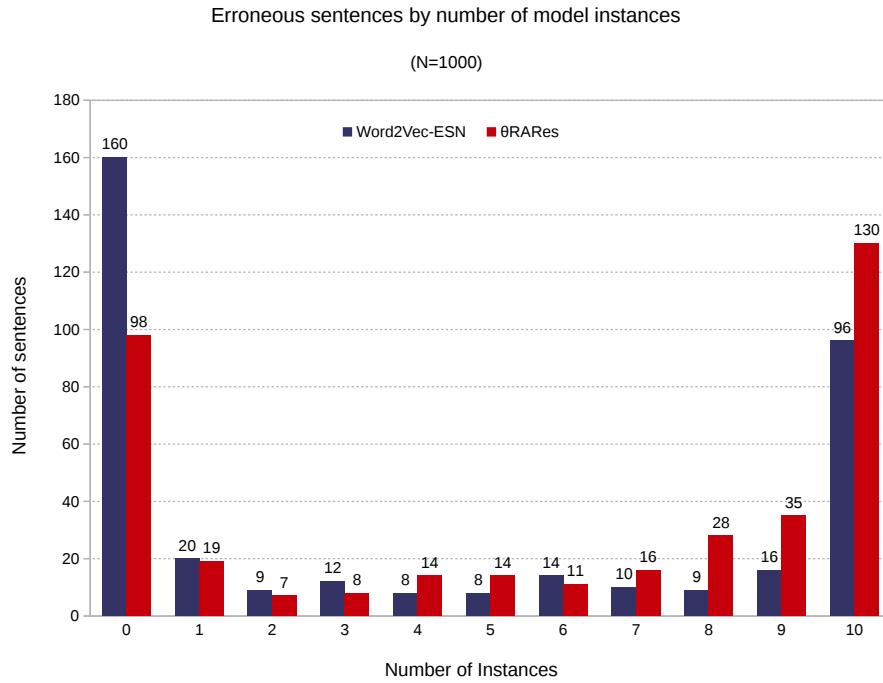


Figure 6.8: Sentence error count by number of model instances with reservoir of 1000 neurons.

With the increase in the reservoir from 500 to 1000 neurons, the number of sentences correctly predicted by all the instances of Word2Vec- θ RAREs and θ RAREs model increased respectively from 146 to 160 and from 35 to 98. Also, 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- θ RAREs model and θ RAREs 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- θ RAREs model is higher than the θ RAREs model. Whereas the count of sentences wrongly predicted by all model instances is lower in Word2Vec- θ RAREs model as compared to that of θ RAREs model.

6.7 Experiment-7: Effect of Word2Vec Word Dimensions

In all the previous experiments, the distributed word vectors of 50 dimensions were used. However, Mikolov et al. [35, 34] 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 of distributed word vector dimensions on the TRA task will be explored.

Word2Vec- θ RARes 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 the TRA task, each of these Word2Vec models was used with the Word2Vec- θ RARes model. For this experiment, the sentences of corpus-373 without the topologically modified coded meaning were used (see fig. 4.2). Simulations were performed using ten model instances each with a reservoir of 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.

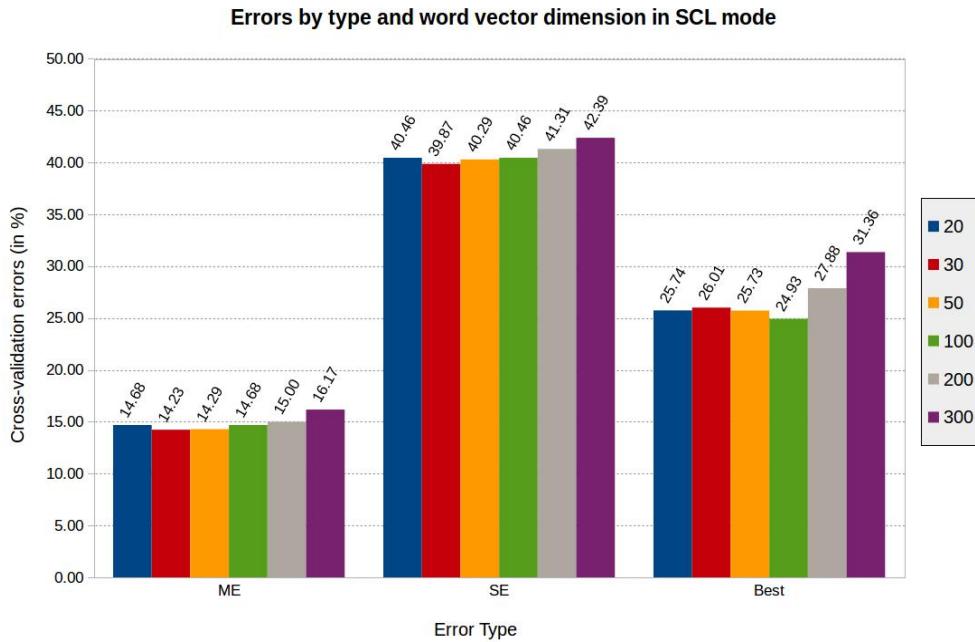


Figure 6.9: **Effect of word vector dimensions on cross validation errors of Word2Vec- θ RARes model in SCL mode:** Cross-validation error are given in percentage. Error type includes ME: Meaning Error, SE: Sentence error, Best: Best generalization error.

Figure 6.9 and 6.10 shows the effect of word vector dimensions on the cross-validation errors of Word2Vec- θ RARes model in SCL and SFL mode respectively. The corresponding errors are also reported in Table A.4. 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 of 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

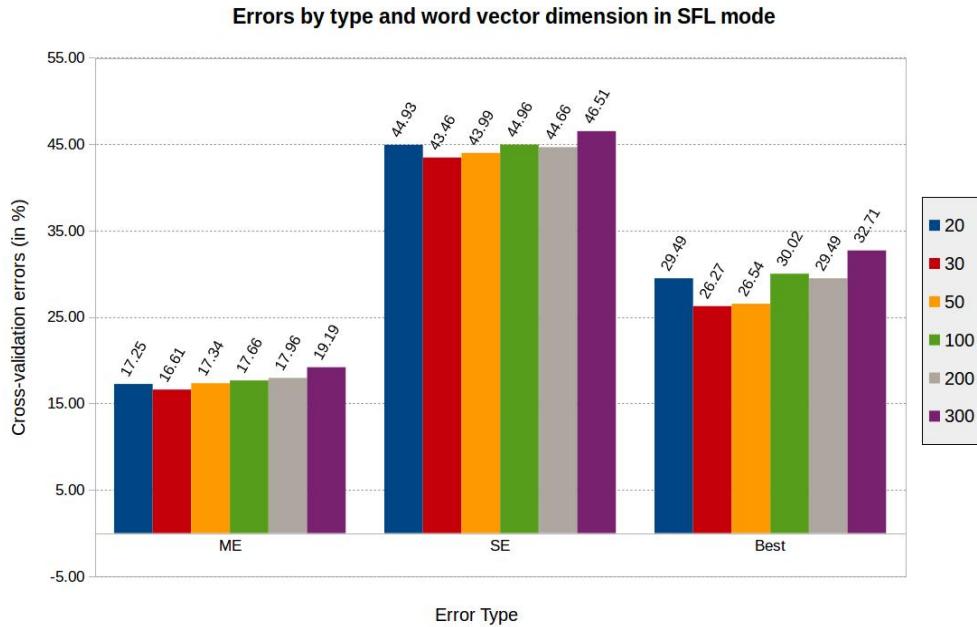


Figure 6.10: Effect of word vector dimensions on cross validation errors of Word2Vec- θ RARes model in SFL mode: Cross-validation errors are given in percentage. Error type includes ME: Meaning Error, SE: Sentence error, Best: Best generalization error.

between SCL and SFL mode was also observed in the previous experiments as well. Overall, it was observed that the word vectors dimensions have a negligible effect on the performance of Word2Vec- θ RARes model for the TRA task.

6.8 Neural Output Activity of the Word2Vec- θ RARes Model

In the previous experiments, we observed that 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- θ RARes model were then analyzed for the sentences in the newly formed corpus of 507 sentences and corpus-373. This analysis of readout activation will allow us to have an insight on how the model 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 the input sentences across the time. The same behavior was also pointed out by Hinaut et. al [21, 20] with θ RARes model.

6.8.1 Output activity for sentences with topologically 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 are active constructions whereas 3 and 4 are passive constructions.

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).

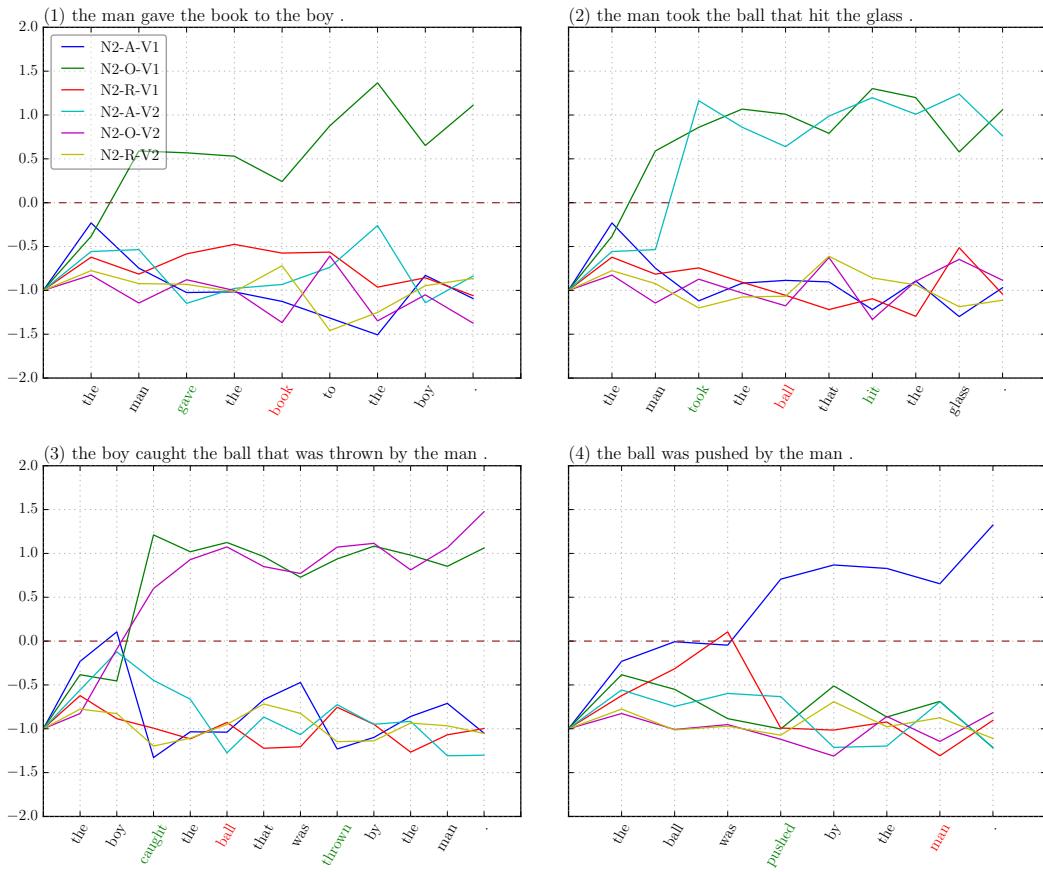


Figure 6.11: **Readout activity of Word2Vec-θRARes model for a sentence with topologically modified coded meaning:** Each coloured line shows the thematic role of Noun-2 (tick marked in red) with respect to Verb-1 and Verb-2 (ticks shown in green).

Each colored line in the graph (see fig. 6.11) represents 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 the first verb (V1).

As all the four sentences start with ‘the’, 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 N2-A-V1 is higher when the first noun (‘ball’) is encountered, but with the arrival of ‘was’ the role of N2 is changed as recipient of V1 (‘pushed’).

In sentence 1, the activation of role N2-A-V1 is maintained, and no other role managed to cross the threshold with the arrival of V1 (‘gave’). Whereas in sentence 2, with the advent of the V1 (‘took’) the activation for N2-A-V2 goes above threshold, indicating the presence of V2 (‘hit’) in the sentence even before the model has encountered the second verb. 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 (‘caught’) and V2 (‘thrown’), i.e., N2-O-V1 and N2-O-V2. In sentence 4, the previously predicted role N2-R-V1 goes below the 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. Similarly, in sentence 2, the predictions of N2 alternates between roles N2-O-V1 and N2-A-V2 but the activation of both the roles remains above the threshold. In sentence 3 the model sees both N2-O-V1 and N2-O-V2 as the preferred predictions, both being almost with similar activations and only alternating slightly. In sentence 4, the prediction of role N2-A-V1 do not change again throughout the sentence ever since the arrival of V1 (‘pushed’).

It can be seen that, at the end of the sentences, the model predicted the meaning of all the four sentences correctly. Also, notice the early prediction of correct roles even before the sentence is completed. The distributed representation of words learned from the nearby words and possibly encapsulating the information about them can be credited for such behavior.

6.8.2 Output activity for sentences without topologically modified coded meaning

Figure 6.12 shows the readout activations of the following two sentences taken from corpus-373.

- (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)*

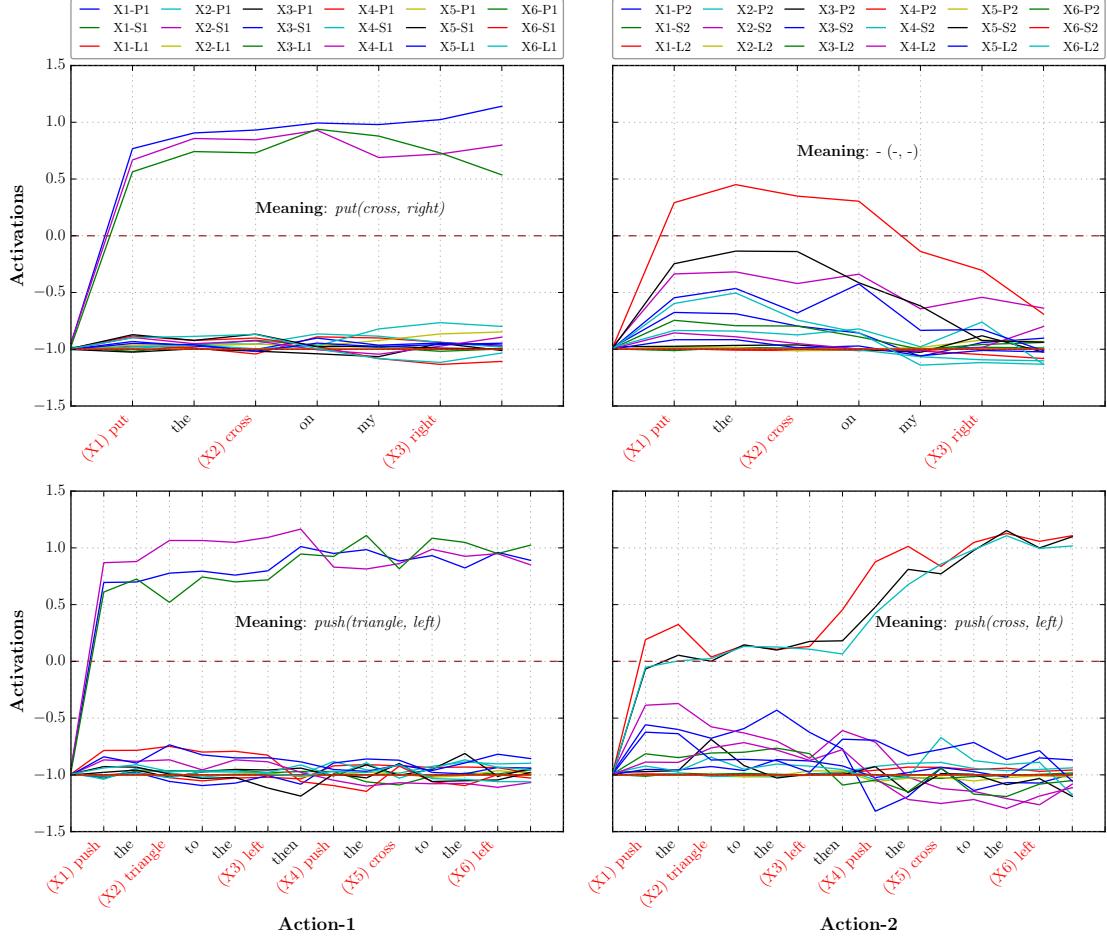


Figure 6.12: **Readout activity of Word2Vec-θRARes model for a simple and complex sentence from corpus-373:** First and second row shows the readout activation for a simple and complex sentence respectively. Each column in the figure shows the activation of the corresponding sentence with respect to an action.

Note that the sentence (A) is a simple instruction to perform only one action and contains three semantic words (X1-X3). Whereas sentence (B), is a complex instruction to perform two actions and contains six semantic words (X1-X6). Each semantic word in the sentences can have a role with respect to a maximum of two actions (see corpus-373 in section 5.1). 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.

The first row of Figure 6.12 shows the readout activity of the model for sentence (A) with respect to action-1 (left) and action-2 (right). 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 (A). Also, for the action-2, notice the online re-analysis behavior of the model where the activation of role (X1-L2) goes above the threshold 0 at the beginning of the sentence but

later drops below the threshold at the end of the sentence.

The second row of Figure 6.12 illustrates 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)*.

6.9 Comparative Analysis of Sentences Meaning

In the previous section, we saw the online re-analysis of sentences with and without the topologically modified coded meanings by the Word2Vec- θ RARes model. So far, we have also seen the performance of Word2Vec- θ RARes model in numbers. In this section, we will analyze for what type of sentences the Word2Vec- θ RARes model correctly predicted the meanings, but the θ RARes model failed. Also, we will see the kind of sentences for which the Word2Vec- θ RARes model failed to predict the meaning correctly. This analysis was done on the results obtained in Experiment-6 using corpus-373.

While analyzing the sentences whose meanings were correctly predicted by all ten instances of the Word2Vec- θ RARes model and incorrectly predicted by the θ RARes model, a significant number of patterns were identified. Almost 85% of the sentences whose meaning were wrongly predicted by all the instances of θ RARes 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 reports the example sentences whose meanings are correctly predicted by all the instances of Word2Vec- θ RARes model and incorrectly predicted by at-least one instance of the θ RARes model. The sentences in the table include only some examples to convey the findings. The Word2Vec- θ RARes model correctly predicted the meaning of the sentences (1) where the color of the objects to be moved is specified (sentences 1-5,11). In other words, the sentences where the adjectives, like ‘red’ or ‘blue’ are used with the semantic words, (2) where the anaphoric reference word ‘it’ (sentences 6-8), pointing to a semantic word within the discourse of sentence is used, (3) where an action to be repeated is specified using the anaphoric words ‘twice’ or ‘two times’ (sentences 9-12) and (4) where a reference location is given with respect to the person instructing the robot (sentences 6, 13-14). Whereas at least one instance of θ RARes model incorrectly predicts the meaning of such sentences.

So far, we have seen the upside of Word2Vec- θ RARes model over θ RARes model, but there are cases where Word2Vec- θ RARes model fails as well. As reported in Table 6.8, 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 the phrase ‘after having’ are predicted wrongly

Table 6.7: Example sentences correctly predicted by all 10 instances of Word2Vec- θ RARes model and mispredicted atleast once by θ RARes 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, -) | - |
| 6 | before grasping the triangle on my left point at it. | point (triangle, -) | grasping(triangle, -) |
| 7 | before grasping the triangle point at it. | point (triangle, -) | grasping(triangle, -) |
| 8 | before hitting the triangle point at it. | point (triangle, -) | hitting (triangle, -) |
| 9 | grasp circle two times | grasp(circle, -) | grasp(circle, -) |
| 10 | point cross two times | point(cross, -) | point(cross, -) |
| 11 | hit twice the blue circle | hit(circle, -) | hit(circle, -) |
| 12 | grasp the circle twice | grasp(circle, -) | grasp(circle, -) |
| 13 | point the circle on my left | point (circle, -) | - |
| 14 | push the cross on my left and then grasp the circle | push (cross, left) | grasp (circle, -) |

by the model (sentences 2-4). Also, the Word2Vec- θ RARes model fails to predict the meaning of grammatically incorrect sentences in most of the cases (sentence 1). It was also noticed that the use of word contractions e.g. ‘youve’, ‘weve’ etc. in the sentences (sentences 5, 6) also leads to an error in meaning prediction by Word2Vec- θ RARes model.

 Table 6.8: Example sentences for which the Word2Vec- θ RARes model failed to predict the meaning correctly.

| 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 θ RARes model which processes the sentences transformed to grammatical form and used the localist word representations to solve the TRA task [20]. We have also seen the proposed neuro-inspired language models (Word2Vec- θ RARes and Word2Vec-ESN classifier) and the experiments performed using them. In this chapter, we finally conclude this research, first by summarizing the findings from the outcome of the experiments performed, then the possible future work identified during the development of this study.

7.1 Conclusion

This thesis started with the goal to test the hypothesis that distributed word representation in combination with a recurrent neural network could be used to achieve better performance on the TRA task. Thus, to test this hypothesis, Word2Vec- θ RARes language model and Word2Vec-ESN classifier were proposed. Several experiments were performed using the proposed Word2Vec- θ RARes 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 in the SCL mode, the proposed Word2Vec- θ RARes model using distributed word embeddings performs better than the θ RARes model which processes the sentences transformed in the grammatical form using localist word vectors.

Performance on a limited number of sentences: It was shown in Experiment-1, that the Word2Vec- θ RARes model learned the sentences thoroughly when trained on limited sentences but failed to generalize on unseen sentences. Whereas, the θ RARes 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 leads us to a conclusion that θ RARes model performs better on small corpus whereas Word2Vec- θ RARes fails to do so.

Performance on corpus-462 and corpus-90582: Recall that the extended corpus-462 and corpus-90582 contain the sentences with the verb inflections (i.e. -ed, -ing, -s). For the same corpora, the θ RARes 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- θ RARes model, the verbs with inflections in the corpora were replaced with the corresponding verb conjugations while preprocessing. The Word2Vec- θ RARes model uses the topologically modified coded meaning for experiments with corpus-462 and corpus-90582. Use of verb inflections as an input to the θ RARes model and topologically modified coded meaning with Word2Vec- θ RARes model do not suffice the requirement of an end-to-end system. Both the θ RARes and Word2Vec- θ RARes model have to depend on a POS tagger 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- θ RARes model generalized on both the corpora but performed much better in the SCL mode when compared to the SFL mode, whereas it is vice-versa in θ RARes model. Additionally, the Word2Vec- θ RARes model outperforms the θ RARes 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- θ RARes and θ RARes model remained almost equivalent. So, it can be concluded that when both the models are dependent on a parser, the Word2Vec- θ RARes model performs better than θ RARes model.

The Word2Vec-ESN classifier also generalizes with high classification scores, when the raw sentences are processed using distributed words embeddings. When the sentences transformed in grammatical form and encoded using localist vector representation are used for processing, the classifier did not generalize well as shown by the low classification scores. Overall the results indicate that the distributed word embeddings improve the performance of the Word2Vec-ESN classifier over localist word vectors.

Performance on corpus-373: As corpus-373 does not contain verb inflections, the θ RARes model processes the sentences without using the input units for verb inflections. Also, the Word2Vec- θ RARes 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 Experiment-6, the Word2Vec- θ RARes model outperforms the θ RARes model. Hence, it can be concluded that the semantic and syntactic information captured in distributed word vectors helps the Word2Vec- θ RARes model to enhance the performance on

the TRA task.

7.1.2 Effect of reservoir and corpus size

It was also shown that the generalization ability of the Word2Vec- θ RARes model on the TRA task also increases with increase in reservoir and corpus size but asymptotes at a point. With the increase in both the reservoir and corpus size, the Word2Vec- θ RARes model generalized better in SCL mode than in SFL mode. However, the generalization ability in both the learning modes becomes stable if the corpus and reservoir size is further increased from the asymptotic point. Overall, the conclusion which can be drawn from this is that with large corpus and reservoir size the Word2Vec- θ RARes model generalizes better. Also, with the increase in corpus and reservoir size the θ RARes model performs better in SFL mode whereas Word2Vec- θ RARes outperforms the θ RARes model in SCL mode.

The Word2Vec-ESN classifier, on the other hand, achieved the high classification score even with the smaller reservoir 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 the TRA task.

7.1.3 Structure of corpus

The results of Experiments 2 and 3, shows that the Word2Vec- θ RARes 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 is scrambled. Thus we can conclude that the Word2Vec- θ RARes 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.

The Word2Vec-ESN classifier, on the other hand, 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- θ RARes 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. Therefore, 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. Hence, two conclusions can be drawn from these results. First, it can be said that the Word2Vec-ESN classifier can perform better with distributed embeddings in the cases where sentences are not completely grammatically correct. Second, it can be considered as a drawback for the TRA task as the model always classifies the argument-predicate pairs correctly even when the sentences do not make any sense.

In 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 abstract 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

Recall that the Word2Vec- θ RARes processes the corpus-373 without using the topologically modified coded meaning. However, it was shown in Experiment-6, that the 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 the TRA task. However, there is a possibility to optimize the performance further on corpus-373 with Word2Vec- θ RARes model by optimizing the ESN parameters on this corpus.

7.1.5 Dimension of word vectors

Mikolov et al. [35, 34] showed that the higher dimensional word embeddings perform better on the word analogy task as compared to word vectors with lower dimensions. We have seen in the result of Experiment-7, that the dimensions of distributed word vectors, do not affect the performance of Word2Vec- θ RARes model significantly on the TRA task. However, the larger word vector dimensions increased the computation cost, but the performance gain was negligible. So, following 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- θ RARes model for the TRA task.

7.1.6 Online re-analysis of sentence meanings

In addition to the generalization on unseen sentences, it was also shown that the proposed Word2Vec- θ RARes 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 an estimated prediction of the sentence meaning for the input given so far [20]. This behavior is not something new. The online re-analysis of sentence meanings was also observed with the θ RARes model. However, the use of word embeddings learned from the context words and possibly encapsulating the information about nearby words enables the Word2Vec- θ RARes model to make a very early prediction of the sentences meaning. 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 listens to the first few

words by making assumptions, gained from 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- θ RARes 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. Therefore, it is not possible for the Word2Vec-ESN classifier to do the online re-analysis of sentence meaning.

7.1.7 Word2Vec- θ RARes vs Word2Vec-ESN classifier

We have seen that the Word2Vec- θ RARes 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 would not be fair to claim which one of them performs better. However, it can be argued that, given a large corpus, both the models perform better when distributed word embeddings are used over localist word vectors. For the TRA task, both models could be employed. Each model has its own advantages and limitations which are stated below:

Input coding: The Word2Vec- θ RARes 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-ESN classifier.

Output coding: The output units of the Word2Vec- θ RARes model encode the thematic roles of semantic words. Thus, in order 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 the number of semantic words and the thematic roles a semantic word can have. The increase in the number of output units also increases the number of learnable reservoir-to-outputs weights (W^{out}) and hence the computational cost also increases. Unlike the Word2Vec- θ RARes model, the number of output units in the Word2Vec-ESN classifier remains independent of the number of semantic words a sentence can have. The number of output units is always equal to the number of thematic roles which a word can have. Therefore, the limited number of output units makes the Word2Vec-ESN classifier computationally cheap.

Reservoir size: The Word2Vec- θ RARes 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- θ RARes model.

Online re-analysis of sentence meaning: As discussed earlier, the Word2Vec- θ RARes have the capability to make online re-analysis of sentence meaning across time. In some cases, the model also predicts the meaning of sentences even before the completion of the sentence. This advantage of online re-analysis and early prediction is not possible with the Word2Vec-ESN classifier.

7.2 Future Work

In this research, the proposed Word2Vec-ESN classifier makes an assumption that the syntactic parser or POS tagger used to identify verbs is 100% accurate. It would be interesting to explore the behavior of the Word2Vec-ESN classifier with the most accurate syntactic parser like Stanford parser [47], Charniak parser [11], Bikel parser [8] or Berkley parser [41].

As described earlier, the Word2Vec model implementation has been modified to update the vocabulary of an already trained model. The new words can be added to the vocabulary, and distributed embeddings can be learned for them. Thus the modified implementation of the Word2Vec model can be extended to make the proposed models, ever-learning systems for the TRA task. However, updating the Word2Vec model with every sentence was also attempted during this research work, but because the computation time was really high and the performance gain was negligible, it was left unexplored for the future work. It would be interesting to explore how the model can be updated in an online manner with the minimal computation time so that it can be feasibly implemented on the robotics platform.

For this work, the combination of the Word2Vec model and ESN was used. 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. However, the dependencies between the thematic roles of words in the sentences were not taken into consideration for learning. To model the conditional probability distribution of the thematic roles of words, Conditional Random Fields (CRF); a log-linear model; could be used [49]. CRFs have been one of the most successful approaches used earlier as well for classification and sequential data labeling tasks [56, 12]. Thus, the Word2Vec-ESN classifier, 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]. It is also important to know that addition of CRF with Word2Vec- θ RARes model does not make any sense because, unlike

Word2Vec-ESN classifier, the model takes the meaning of the whole sentence as teacher output to learn thematic roles.

Distributed word vectors obtained by training the Word2Vec model on one language corpus (e.g., English) can also be translated to get the most similar word in any other target language (e.g., French). This translation is achieved by linearly projecting (rotation and scaling) the word vectors of source language on the target language [36]. Therefore, the Word2Vec- θ RAsRes language model can also be investigated further for multiple language acquisition [22].

Appendix A

Complete Simulation Results

Table A.1: Effect of reservoir size on Word2Vec- θ RARes model in different learning modes.

| Word2Vec- θ RARes Model | | | | | |
|--------------------------------|------|----------|-------|----------|-------|
| Reservoir size | | SCL mode | | SFL mode | |
| | | ME | SE | ME | SE |
| 90 | mean | 38.30 | 92.52 | 46.41 | 98.09 |
| | std | 1.53 | 1.67 | 1.02 | 1.29 |
| 291 | mean | 17.68 | 60.26 | 22.00 | 69.13 |
| | std | 0.80 | 1.74 | 0.97 | 3.03 |
| 493 | mean | 11.63 | 40.34 | 13.60 | 45.61 |
| | std | 0.65 | 2.58 | 0.64 | 2.14 |
| 695 | mean | 9.76 | 32.56 | 10.64 | 34.83 |
| | std | 0.61 | 1.56 | 0.58 | 1.42 |
| 896 | mean | 8.39 | 25.56 | 9.79 | 28.70 |
| | std | 0.32 | 0.73 | 0.50 | 0.60 |
| 1098 | mean | 7.98 | 22.17 | 9.41 | 26.09 |
| | std | 0.56 | 1.45 | 0.26 | 1.14 |
| 1776 | mean | 7.57 | 19.65 | 9.79 | 24.70 |
| | std | 0.39 | 0.84 | 0.43 | 0.90 |
| 2882 | mean | 6.99 | 18.04 | 10.85 | 26.22 |
| | std | 0.55 | 1.00 | 0.90 | 1.61 |
| 3988 | mean | 7.20 | 17.74 | 11.02 | 26.17 |
| | std | 0.48 | 0.45 | 0.50 | 1.63 |
| 5094 | mean | 7.07 | 17.83 | 11.06 | 25.87 |
| | std | 0.53 | 1.54 | 1.08 | 1.34 |

Meaning (ME) and Sentence error (SE) in different learning modes with Word2Vec- θ RARes model. The errors are given in percentage and averaged over several model instances. The reservoir size is shown in first column. SCL: Sentence Continuous Learning; SFL: Sentence Final Learning; std: standard deviations. Simulations were performed with 5 model instances. Model was tested using 10-fold cross-validation approach.

Table A.2: Effect of reservoir size on Word2Vec-ESN classifier in two different configurations.

| Reservoir size | | Word2Vec-ESN Classifier | | | | | |
|----------------|------|-------------------------|-------|-------|-----------------|-------|-------|
| | | Configuration-1 | | | Configuration-2 | | |
| | | Pr | Re | F1 | Pr | Re | F1 |
| 50 | mean | 83.09 | 52.59 | 61.12 | 61.78 | 46.31 | 51.23 |
| | std | 4.57 | 7.76 | 8.10 | 1.00 | 4.05 | 3.11 |
| 100 | mean | 92.47 | 84.67 | 88.05 | 61.90 | 61.26 | 60.72 |
| | std | 1.14 | 1.87 | 1.37 | 0.19 | 1.22 | 0.58 |
| 250 | mean | 96.35 | 91.35 | 93.58 | 61.84 | 65.24 | 62.51 |
| | std | 0.39 | 0.25 | 0.18 | 0.10 | 0.84 | 0.36 |
| 400 | mean | 97.04 | 91.62 | 94.03 | 61.87 | 65.54 | 62.67 |
| | std | 0.34 | 0.24 | 0.19 | 0.05 | 1.43 | 0.62 |
| 600 | mean | 96.88 | 91.77 | 94.05 | 62.00 | 66.98 | 63.21 |
| | std | 0.13 | 0.13 | 0.10 | 0.08 | 0.54 | 0.24 |
| 800 | mean | 96.69 | 91.83 | 94.00 | 61.94 | 67.10 | 63.30 |
| | std | 0.19 | 0.07 | 0.10 | 0.04 | 0.51 | 0.26 |
| 1050 | mean | 96.73 | 91.80 | 94.01 | 62.03 | 67.77 | 63.54 |
| | std | 0.13 | 0.03 | 0.07 | 0.03 | 0.29 | 0.17 |
| 1600 | mean | 96.57 | 91.70 | 93.88 | 61.92 | 68.37 | 63.85 |
| | std | 0.17 | 0.10 | 0.13 | 0.02 | 0.57 | 0.23 |
| 2250 | mean | 96.29 | 91.52 | 93.62 | 61.82 | 68.33 | 63.82 |
| | std | 0.13 | 0.05 | 0.09 | 0.07 | 0.83 | 0.37 |
| 3320 | mean | 96.05 | 91.38 | 93.46 | 61.80 | 68.24 | 63.83 |
| | std | 0.22 | 0.19 | 0.21 | 0.08 | 0.94 | 0.39 |
| 3860 | mean | 96.07 | 91.39 | 93.47 | 61.85 | 67.74 | 63.57 |
| | std | 0.22 | 0.11 | 0.15 | 0.05 | 2.80 | 1.19 |
| 4500 | mean | 95.91 | 91.28 | 93.34 | 61.89 | 70.01 | 64.42 |
| | std | 0.25 | 0.03 | 0.13 | 0.07 | 3.60 | 1.51 |

Precision (Pr), Recall (Re) and F1-Score (F1) in different configuration of Word2Vec-ESN classifier. The scores are given in percentage and averaged over several model instances. Configuration-1: raw sentences are processed, and distributed vectors of the argument-predicate pairs are input to ESN. Configuration-2: Sentences transformed to grammatical form and localist word vectors are input to ESN. std: standard deviations. Simulations were performed with 5 model instances. Model was tested using 10-fold cross-validation approach.

Table A.3: Effect of sub-corpora size on Word2Vec- θ RARes model in different learning modes.

| sub-corpora size | | Word2Vec- θ RARes Model | | | |
|------------------|------|--------------------------------|-------|----------|-------|
| | | 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- θ RARes 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 performed with 5 model instances each having a reservoir of 1000 neurons. Model was tested using 10-fold cross-validation approach.

Table A.4: Effect of word vector dimensions on Word2Vec- θ RARes model in different learning modes.

| Word2Vec- θ RARes Model | | | | | | |
|--------------------------------|------|----------|-------|-------|----------|-------|
| | | SCL mode | | | SFL mode | |
| | | ME | SE | Best | ME | SE |
| 20 | mean | 14.68 | 40.46 | 25.74 | 17.25 | 44.93 |
| | std | 0.58 | 1.58 | - | 0.76 | 1.36 |
| 30 | mean | 14.23 | 39.87 | 26.01 | 16.61 | 43.46 |
| | std | 0.53 | 0.57 | - | 0.67 | 0.70 |
| 50 | mean | 14.29 | 40.29 | 25.73 | 17.34 | 43.99 |
| | std | 0.34 | 1.13 | - | 0.36 | 1.38 |
| 100 | mean | 14.68 | 40.46 | 24.93 | 17.66 | 44.96 |
| | std | 0.58 | 1.58 | - | 0.55 | 1.66 |
| 200 | mean | 15.00 | 41.31 | 27.88 | 17.96 | 44.66 |
| | std | 0.73 | 1.55 | - | 0.63 | 1.19 |
| 300 | mean | 16.17 | 42.39 | 31.36 | 19.19 | 46.51 |
| | 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- θ RARes 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 instances each having a reservoir of 1000 neurons. Model was tested using LoO cross-validation approach.

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] Yinyin Liu, Janusz A Starzyk, and Zhen Zhu. Optimized approximation algorithm in neural networks without overfitting. *IEEE Transactions on Neural Networks*, 19(6):983–995, 2008.
- [33] Mantas Lukoševičius. *A Practical Guide to Applying Echo State Networks*, pages 659–686. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [34] T Mikolov and J Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.
- [35] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [36] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

- [37] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2009.
- [38] 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.
- [39] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.
- [40] Jacob Persson, Richard Johansson, and Pierre Nugues. Text categorization using predicate–argument structures. *Proe. the*, 1:142–149, 2008.
- [41] 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.
- [42] 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.
- [43] 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.
- [44] 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>.
- [45] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.
- [46] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.
- [47] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *ACL (1)*, pages 455–465, 2013.

- [48] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.*, 45(4):427–437, July 2009.
- [49] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *arXiv preprint arXiv:1011.4088*, 2010.
- [50] Ciza Thomas and N. Balakrishnan. Improvement in minority attack detection with skewness in network traffic, 2008.
- [51] 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.
- [52] 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.
- [53] 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.
- [54] 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.
- [55] 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.
- [56] 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

