Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Word2Vec and Echo State Network For Thematic Role Assignment

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

Department Informatik
MIN-Fakultät
Universität Hamburg

Submitted by
Surender Kumar
on
30.04.2013

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

Surender Kumar
Matriculation Number: 6519753
Kaemmererufer 13
22303 Hamburg

Abstract

Humans have a remarkable capability of acquiring language and in particular more than one languages. More interestingly they learn it within the same neural computing substrate. But how does the structure of a sentence is mapped to its meaning within the brain is still an open issue? To understand this process of language acquisition, several neural language models were proposed. Most of the existing model are either using syntactic parse trees to create handcrafted features or using localist vector representation of words to process the sentences. Hinaut et al. [47] proposed a $\theta RARes$ neural language acquisition model, for thematic role assignment which learned the thematic roles of the input sentences purely from the grammatical structure of the sentences. They also treated the words in the sentence as discrete atomic symbols which do not carry any semantic information about the words i.e. localist word representation was used. This thesis, proposes an end-to-end neural language model, in an extension to $\theta RARes$ model, which takes into account the semantics of the words and temporal aspect of the input sentences. The model contains a word2vec unit, which generates the distributed vector representation of words which captures the semantic and syntactic relationships along with a recurrent neural network namely, Echo state Network, with fixed random connections, which is capable of modeling the sequential data. Thus the model takes a raw sentence as an input which is processed word by word and as an output, the meaning of an input sentence is obtained. Our results shows that the proposed model performs better than the $\theta RARes$ model on thematic role assignment task.

Zusammenfassung

Hier die deutsche Zusammenfassung einfügen (notwendig).

Contents

1 Introduction **1**
- 1.1 Previous Work . . . 1
- 1.2 Motivation and Hypothesis . . . 2
- 1.3 Proposed Models . . . 3
- 1.4 Outline . . . 4

2 Basics of Word2Vec and Echo State Network **5**
- 2.1 Word2Vec . . . 5
  - 2.1.1 CBOW Model . . . 5
  - 2.1.2 Skip-gram Model . . . 8
  - 2.1.3 Properties of Word2Vec embeddings . . . 9
- 2.2 Echo State Network (ESN) . . . 11
  - 2.2.1 ESN Architecture . . . 11
  - 2.2.2 Training ESN . . . 12

3 Related Work and Open Issues **15**
- 3.1 Overview of $\theta RARes$ Model . . . 15
  - 3.1.1 Limitation of $\theta RARes$ model . . . 17
  - 3.1.2 Research Hypothesis . . . 20

4 Approaching Word2Vec-ESN Language Model **21**
- 4.1 Word2Vec-ESN Language Model . . . 21
  - 4.1.1 Model initialization . . . 21
  - 4.1.2 Training model . . . 22
  - 4.1.3 Evaluation Metrics . . . 24
- 4.2 Variant of Word2Vec-ESN Model . . . 25
  - 4.2.1 Training model variant . . . 26
  - 4.2.2 Decoding Output . . . 27
  - 4.2.3 Evaluation Metrics . . . 27

5 Experiments **29**
- 5.1 Corpora and pre-processing . . . 29
  - 5.1.1 Corpora For TRA Task . . . 29
  - 5.1.2 Corpus For Training Word2Vec Model . . . 31
- 5.2 Experimental Setup . . . 32

5.2.1 Obtaining Word Embeddings 32
5.2.2 ESN reservoir initialization 33
5.2.3 Learning ESN parameters 34
5.2.4 Input and Ouput Coding 35
5.3 Experiments 35
5.3.1 Experiment-1: Model performance on small corpous 36
5.3.2 Experiment-2: Generalization Capabilities 36
5.3.3 Experiment-3: Effect of Corpus structure 37
5.3.4 Experiment-4: Effect of Reservoir size 38
5.3.5 Experiment-5: Effect of Corpus size 38
5.3.6 Experiment-6: Generalization on new corpus 39
5.3.7 Experiment-7: Effect of Word2Vec word dimensions 39

6 Results and Discussion **41**
6.1 Results and Discussion 41
6.1.1 Experiment-1: Model performance on a small corpous 41
6.1.2 Experiment-2: Generalization Capabilities 41
6.1.3 Experiment-3: Effect of Corpus structure 44
6.1.4 Experiment-4: Effect of Reservoir size 45
6.1.5 Experiment-5: Effect of Corpus size 47
6.1.6 Experiment-6: Generalization on new corpus 48
6.1.7 Experiment-7: Effect of Word2Vec word dimensions 49
6.1.8 Neural output activity of the Word2Vec-ESN model 49

7 Conclusion And Future Work **55**
7.1 Conclusion 55
7.2 Future Work 55

A Nomenclature **57**

B Complete Simulation Results **59**

Bibliography **63**

List of Figures

2.1 The CBOW model 6
2.2 The Skip-gram model 7
2.3 Word2vec semantic regularities 9
2.4 Word2Vec word clustering 10
2.5 Word2Vec language translation property 10
2.6 Architecture of classical ESN 12

3.1 Word2vec semantic regularities 16
3.2 Word2vec semantic regularities 16
3.3 Different type of sentence structure to meaning relations 18

4.1 Architecture of Word2Vec-ESN model 21
4.2 Neural comprehension of Word2Vec-ESN Model 22
4.3 Neural comprehension of Word2Vec-ESN Model 24
4.4 Variant of Word2Vec-ESN language model 26

6.1 Normalized confusion matrix on corpus 462 with Word2Vec-ESN model variant 43
6.2 Effect of reservoir size on Word2Vec-ESNM model 46
6.3 Effect of reservoir size on Word2Vec-ESN model variant 46
6.4 Effect of corpus size on Word2Vec-ESN model 47
6.5 Effect of reservoir size on Word2Vec-ESN model variant 48
6.6 Effect of word vector dimensions on Word2Vec-ESN model 50
6.7 Effect of word vector dimensions on Word2Vec-ESN model 50
6.8 Online re-analysis of activation by Word2Vec-ESN Language model: 52
6.9 **Effect of corpus size on cross validation errors using localist word vector as reported in :** Description goes here. 53

List of Tables

3.1 Localist vector representation of sentence 16

6.1 Mean and standard deviation of meaning and sentence error on train and test set of coprus-462 in different learning modes. 42
6.2 Classification scores produced by Word2Vec-ESN model variant on corpus-462 in two configurations. 43
6.3 Training and testing classification scores for individual roles when using Word2Vec-ESN model variant in two different configurations. 45
6.4 Word2Vec-ESN model generalizing on new coprus 49

B.1 Effect of sub-corpora size on Word2Vec-ESN model in different learning modes. 60
B.2 Effect of word vector dimensions on Word2Vec-ESN model in different learning modes. 61

Chapter 1

Introduction

Thematic Role Assignment (TRA) is a supervised learning problem which aims to identify events and its participants from a sentence and determine *"Who did what to whom"*. In other words assigning roles to words (arguments) in a sentence with respect to a verb (predicate). The role typically includes 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 terminology (NLP) the problem is studied under the name of Semantic Role Labelling (SRL). Hence, TRA or SRL is a form of simplistic semantic parsing which aims to determine the predicate-argument structure for a verb in the given sentence [47]. Understanding the semantics of the text plays an important intermediate step in a wide range of real-world applications such as machine translation [26], information extraction [4], sentiment analysis [46], document categorization [34], human-robot interaction [16, 4] etc.

1.1 Previous Work

Many successful traditional systems consider SRL as a multiclass classification problem and uses linear classifier such as Support Vector Machines (SVM) to tackle the problem [25, 36, 35]. These systems were based on pre-defined feature templates derived from syntactic information obtained by parsing and producing parse trees of the sentences in the training corpus. However, in an analysis, it was found that the use of syntactic parser certainly leads to degradation of thematic roles predictions [35]. 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 selcted features. The feature templates are often required to be re-designed when the task is to be performed in different languages, corpus or when the data distribution is changed [47].

In order to avoid manual feature engineering, SRL task was also attempted with neural network models. Collobert et al. [12] first attempted to build an end-to-end system without using parsing and using a neural model with word

embeddings layer, Convolutional Neural Network (CNN) and Conditional Random Field (CRF). The model was less successful as CNN cannot employ long-term dependencies within a sentence since it can only take into account the words in limited context [47]. However, to increase the model performance they also resorted to using syntactic features by using parse trees of charnink parser [9].

Recurrent Neural Networks (RNN) has also been used for wide range of NLP task and also recently with Echo State Network (ESN), a variant of RNN. The tasks used were diverse from predicting next word given the previous words to learning grammatical structures [43]. RNN makes use of sequential information and acts as a memory unit and captures the information processed in the past [13]. The ESN have several advantages over simple RNN. First, ESNs are capable of modeling long-term dependencies in the sentence. Second, while processing long sequence the gradient parameter vanishes or explodes in simple RNNs [6]. Third, unlike simple RNN, ESNs are computationally cheap as the recurrent layer (reservoir) in ESN is randomly initialized and only the connections from recurrent layer to read-out layers are learned [23, 27]. These advantages of ESN over RNN makes it a good choice to be used for TRA task.

Hinaut et al. [15] proposed a generic neural network architecture, $\theta RARes$ model, using Recurrent Neural Network based on reservoir computing approaches, namely Echo State Network to solve TRA task. The proposed architecture models the language acquisition in the brain and provided a robust and scalable implementation on robotics architecture [15, 16]. The model is based on the notion of grammatical construction: mapping of word order (surface form) to its meaning. They first transformed the raw sentences by replacing the semantic words (nouns, verbs etc.) with a unique token 'SW' then the sentences are input to model sequentially, word by word across time along with the coded meaning (i.e. thematic roles of semantic words) of the input sentence for training. The model learns the thematic roles of all the semantic words in the input sentence during training. During testing, the model predicts the coded meaning of the previously unseen sentences. See chapter 3 for more details about $\theta RARes$ model.

1.2 Motivation and Hypothesis

Like many other traditional NLP systems, Hinaut et al. [15] also treated words as discrete atomic symbols and used localist vector representation of words as an input. Treating each word as a discrete symbol does not provide any relational information to the model which may exist between two words. For example, if words *'pink'* and *'red'* are represented using localist representation with vectors [1,0] and [0,1] respectively, the semantic relationship (i.e. both are colors) between these two words is lost [19]. Although, replacing the semantic words with 'SW' token makes it possible to train the model on a small corpus as the 'SW' token can be replaced with different semantic words (nouns, verbs etc.) to form a sentence. Whereas, the 'SW' token in itself does not carry any semantics and thus does not allow the model to take into account the semantics of the words. This makes it

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

Motivated by the limitations of localist representation of words and transformation of raw sentences into its abstract form by replacing semantic words with 'SW' token described above, this work hypothesize that the use of distributed word representations, which can capture the syntactic and semantic relationship of words, could possibly improve the performance of the model on TRA task. One such model for learning distributed word embeddings was proposed by Mikolov et al. [28] widely known as the Word2Vec model. Word2Vec model learns high quality, low-dimensional vector representation of words from a large corpus in an unsupervised way. The resulting word vector of this model encodes the semantics of the words. The model learns the word embeddings by taking into account the context (neighbouring) words. The obtained word embeddings captures several language regularities and patterns [30, 28] and can be observed by performing the linear operations on the word vectors. For example, *vector('king') - vector('man') + vector('woman')* $\approx$ *vector('queen')*. Unlike other neural network models [33, 11, 44, 31] for obtaining word embeddings, training Word2Vec is computationally cheap and efficient [28]. Training of word2vec model and properties of resulting distributed word embeddings will be discussed later in more detail in Chapter 2.

1.3 Proposed Models

This work, proposes an end-to-end neural language model for thematic role assignment. The model is named as *Word2Vec-ESN* language model. The Word2Vec-ESN 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 use of the model for TRA task. The model receives the raw sentences sequentially and process a sentence word by word generating. The word2vec unit being the first unit in the model, takes the input word to generate the distributed word embeddings. The generated word vector by word2vec model can then be used by ESN for learning thematic roles of the input sentences. Note that the proposed model is an extension of $\theta RARes$ model [15], where unlike the latter the raw sentences are not transformed to grammatical forms and word2vec word vectors are used over localist word representation as an input to ESN.

Apart from Word2Vec-ESN model, this thesis also propose a variant of Word2Vec-ESN language model which only differs from the original in the way, the sentences are processed and results are evaluated. Thus in this model variant, the inputs and outputs of the model are changed. The input feature to this model variant is the current word and the predicate, with respect to which it is processed. The output units encode the possible role (e.g. Predicate, Agent, Object, Recipient and No Role) of the input argument-predicate pairs, unlike the original model where output units encode the possible thematic roles of all semantic words in the input sentences. For the evaluation of this model variant, the metrics proposed for Conll-2004 [**?**, 8] and CoNLL-2005 [7] SRL task is used. Both Word2Vec-ESN

language model and its variant are discussed in more detail in Chapter 4.

There are several neural network based model to obtain the word the word embeddings [33, 11, 44, 31] but a systematic comparison of them on TRA task is beyond the scope of this work. To this date, there is no research conducted with this combination of word2vec model and ESN. This also makes this study novel.

1.4 Outline

This thesis is organized as follows. The Chapter 2, describes the basics of word2vec model and echo state network model. The chapter also describe in detail the training of word2vec model and properties of resulting word vectors. Also, this chapter describes training and control parameters of ESN. Chapter 3 looks upon related work with primary focus on neural network $\theta RARes$ language model, its limitations, the motivation and hypothesis for the current work. Subsequently chapter 4 proposes the Word2Vec-ESN model and its variant. This chapter also describes the implementation and the metrics used to evaluate the performance of the proposed model and its variant. Chapter 6 first describes the corpora used and then the experimental setup along with the experiments performed using the proposed model. We will then see the results obtained using proposed model on TRA task. In this chapter, we also compare the results of Word2Vec-ESN model with the results obtained from $\theta RARes$ model and analyze them. Finally, in the chapter 7 we conclude the study and the possible future work.

Chapter 2

Basics of Word2Vec and Echo State Network

2.1 Word2Vec

Word2vec is a neural probabilistic language model based on Distributional Hypothesis which states that the words that appear in the same context share the semantic meaning [19]. The proposed by Mikolov et al. [28], which takes in input a large text data to generate the distributed word embedding of the words present in the text and also preserve the linear regularities among words. In other words, it maps the words into a continuous vector space where semantically related words are placed close to each other in the vector space. Earlier the words have been treated as discrete atomic symbols in all traditional NLP system, where the words were represented in a localist fashion. Localist representation of words does not contain any semantic or syntactic information about the word it is encoding and thus depriving the NLP systems to utilize this information while processing [3]. Word2vec neural word embeddings overcome this issue and capture the semantics and syntactic information of the word in a computationally-efficient manner [29]. For learning word embedding two neural architecture were proposed, the Continuous Bag Of Word (CBOW) and Skip-Gram (SG) [29, 28]. Both the models are architecturally same i.e. both have three layers, the input layer, hidden layer and an output layer, but have different training objectives. The architecture of both CBOW and SG models is shown in figure 2.1 and 2.2 respectively. In the next sections, we give a brief overview of CBOW model and SG model. However, we used the skip-gram model in our work because it was shown that it produce better word-embeddings as compared to CBOW [28].

2.1.1 CBOW Model

CBOW model is a three layered neural model with the training objective to predict a target word (e.g. Peter) given some context words (John gave the ball to). Figure 2.1 shows the CBOW network architecture with a simplified case of one context word. The model is trained on the dataset having a vocabulary size V

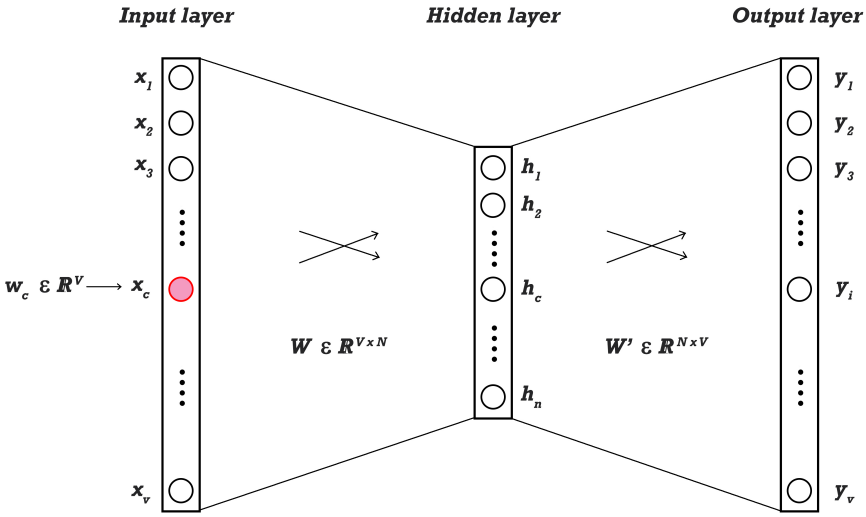

Figure 2.1: **The CBOW model:** In the CBOW model, the objective is to predict the target word from the words in the context (or neighboring words). The context words are input to the model (in this case only one) using localist representation where only one vector element corresponding to input word is active (x_c, shown in red). The model outputs the probability for each word in vocabulary, which is maximized for actual target word during training. Adapted from [38].

in an unsupervised way to achieve the objective. The hidden layer is of size N, the dimensions of desired word embedding, and neurons in both the adjacent layer i.e. input and output layers, are fully connected to hidden layer neurons. The input to the network is the context word $w_c \in \mathbb{R}^V$ and is represented using localist representation where only one unit x_c at index c will be 1 out of V units $w_c = [x_1, ..., x_V]$ and all other units will be 0 [38]. The activation of hidden layer is then given by :

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

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

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

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

where u_i gives the score of i^{th} word,w_i, for i = 1, 2...V. $W^{'} \in \mathbb{R}^{N \times V}$ is the weight matrix between hidden and output layer. $W_i^{'}$ is the i^{th} column vector of matrix

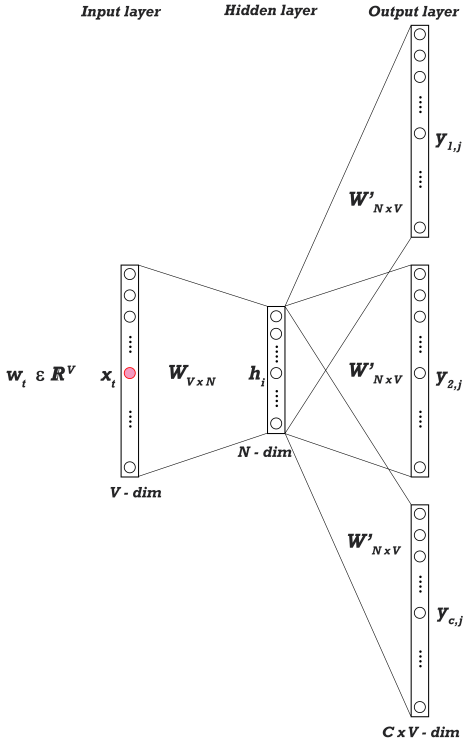

Figure 2.2: **The skip-gram model:** In the skip-gram model, the objective is the reverse of CBOW. It predicts the context words from a target word. The target word is input to the model using localist representation where only one vector element corresponding to input word is active (x_t, shown in red). The model then maximizes the probability of context words during training. Adapted from [38].

$W^{'}$. The computed scores are then converted to posterior probabilities by output neurons with softmax activation function. Thus aliasing $W_i^{'}$ as $v_i^{'}$ we get output probabilities as:

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

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

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

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

In case of multiple context words is input to the network, the equation 2.1.1 only change to:

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

where k is the size of context window. This equation averages vector embeddings of all context words [38].

2.1.2 Skip-gram Model

In the Skip-Gram model training objective is reversed from that of CBOW model. In other words, the objective is to learn the vector representation of the word that is good in predicting the context words [28]. Thus for a given sequence of words $\{w_1, ...w_V\}$, the objective is to maximize the average log probability.

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

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

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

where v'_{t+j} and v_t are the vector representation of word w_{t+j} and w_t respectively.

The objective function is thus optimized using stochastic gradient descent to learn the good word vectors. Calculating the full softmax is computationally expensive as it needs to compute and normalize probability for every other word w in the vocabulary V for a given input word (w_c for CBOW or w_t for skip-gram) at every training step. Thus negative sampling was proposed for learning the word embeddings[28].

Skip-gram with negative sampling

For learning word features full probabilistic models was not required. So in skip-gram negative sampling is used for approximation of word features. This technique treats feature learning as a binary classification (logistics regression) problems [28, 19]. The model is thus trained to distinguish the target word from k imaginary noise words w_{noise}, in the same context. Thus the log probability $p(w_{t+j}|w_t)$ in equation 2.1.8 is now approximated by:

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

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

In order to accelrate the learning and improve the quality of words vectors, the most frequently occuring words in the training corpus can be subsampled. A large training corpus, usually contains some words like "a", "the", "in" etc. which occurs million of times. The model is less benifitted from the most frequent words as compared to the words which occure rarely. For example the co-occurence of "man" with "the" is less importatnt as compared of co-occurence of "man" with "woman", as the word "the" occures with most of words as well [28]. So in order to reduce the effect of most frequent words, each word (w_t) in the vocabulary of training corpus was discarded with a probability:

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

where $f(w_t)$ is the frequency of word w_t and t is a threshold. Using this formula, the frequencies of the words are preserved while all the words with frequency greater than t are subsample [28].

2.1.3 Properties of Word2Vec embeddings

Although the word2vec model is simple in architecture and easy to train, it produces word vector embedding which surprisingly encodes several linguistic regularities and patterns [30, 28]. More importantly, it is astonishing because the network was not explicitly trained for these linguistic properties (see fig. 2.3 and 2.5). The distributed word embeddings encode semantic and syntactic properties of the words as a constant vector offset between a pair of words sharing a specific relationship[28]. For example, the word embeddings "$King - Queen \approx man - woman$", "$apples - apple \approx cars - car$", "$walking - walked \approx swimming - swam$".

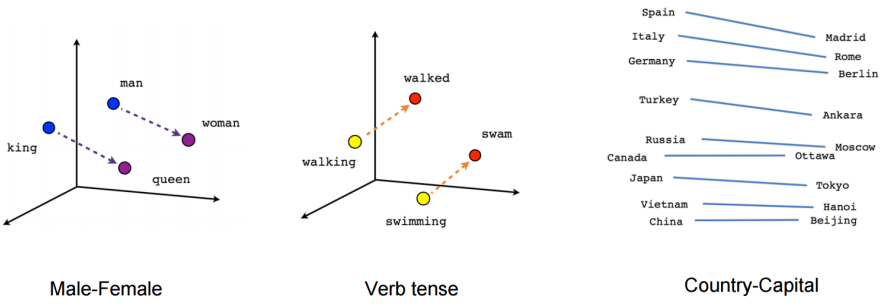

Figure 2.3: **Word2vec semantic regularities.**

[2]

The another interesting property is that the semantically related words are placed close to each other in word vector space, thus forming clusters of semantically related words. It was also observed the word embeddings of similar words in different languages have the same geometrical arrangement in embedding space

[2]https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

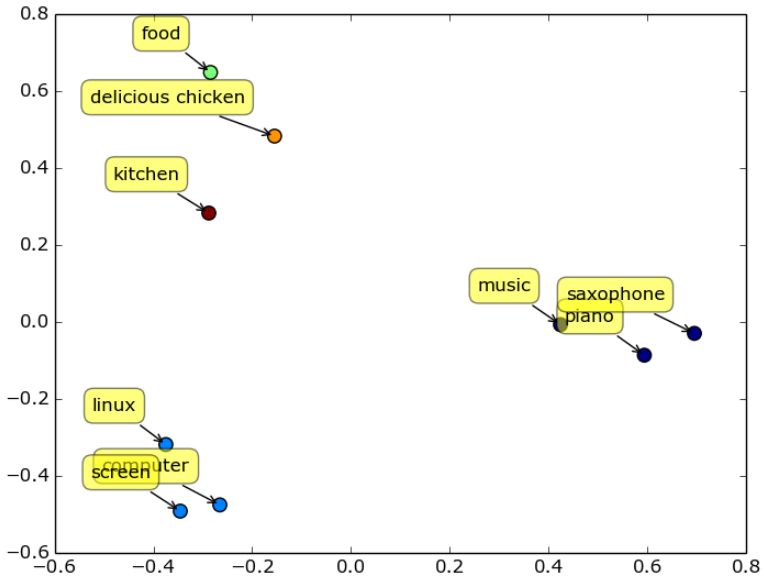

Figure 2.4: **Word Clustering with word2vec word embedding:** The figure shows the word clustering property obtained by projecting the word vectors on two dimesional space using PCA. The words vector taken from pretrained word2vec Google News corpus[1]

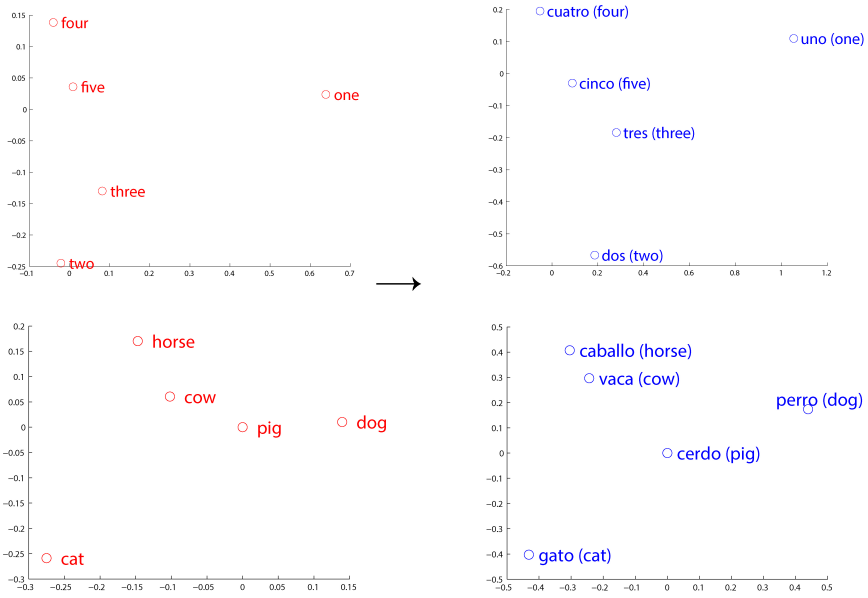

Figure 2.5: **Word2Vec language translation property.**

of respective language. Thus it is also possible to learn the linear mapping between different embedding space by vector rotation and scaling [30]. Several other regularities can also be captured by performing the basic linear operation on word-embeddings [28].

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 Herbert Jaeger [22] and is believed to closely resemble the learning mechanism in biological brains. ESN is found to be computationally simple and inexpensive to process the temporal or sequential data. The main idea of ESN is to operate the random, large, fixed RNN with the input signal and the non-linear response generated by each neuron of the RNN is collectively combined with the desired output signal using regression to learn the output weights [21, 22, 20].

2.2.1 ESN Architecture

ESN is surprisingly efficient variant for RNN training (see fig. 2.6). In the standard RNN, all the weights are required to be tuned, even though it was shown that RNN works well enough even without full adaptation of weights. The classical ESN mainly contains three layers, input layer, the hidden layer (also known as the reservoir) and the readout layer. The input layer is fully connected to the hidden layer and both the hidden layer and the input layer is connected to the output layer. The output layer is fully connected back to the hidden layer. However, the connection from input to output layer and output layer to hidden layer is optional and depends on the task.

The weights from input to reservoir (i.e. W^{in}) and from the reservoir to reservoir (i.e. W^{res}), are sparsely and randomly initialized and more crucially remains untrained during training. The non-zero element in sparse input weight matrix W^{in} and reservoir weight matrix W^{res} are generated from uniform or normal distribution. The weights from the reservoir to output layer (i.e. W^{out}) are the only weights learned during supervised training [22, 27]. For ESN approach to work the reservoir should possess the Echo State Property: 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 [27, 21].

To ensure the echo state property in ESN, firstly, the reservoir weights matrix W^{res} and the input weights matrix W^{in} are often generated sparsely (i.e. most of the elements in these matrices will be zero) and randomly from a normal or uniform distribution [27]. The input weight matrix is, however, a bit denser than the reservoir weight matrix. The sparsely generated random reservoir weights matrix W^{res} is often scaled such that its spectral radius $\rho(W^{res})$ i.e. largest absolute

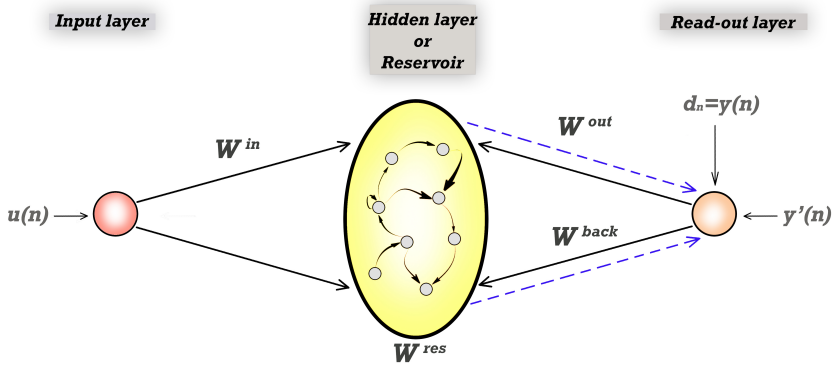

Figure 2.6: **Architecture of classical ESN:** The reservoir is the recurrent neural network with N_x units and initialized with sparse and random connections. The reservoir is provided with an input sequence $u(n)$ from the input layer and the teacher signal $y(n)$ from to output layer during training. The input to reservoir weights (W^{in}), output to reservoir weights (W^{back}, optional and depends on task) and reservoir to reservoir weight (W^{res}) are randomly initialized and stays static during learning. The output weight from the reservoir to output unit are the only weights learned by the network during training. Adapted from [27]

eigenvalue, is less than one. To scale the randomly generated W^{res} matrix, it is first divided by its spectral radius and then multiplied with desired spectral radius [21].

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

where W^{res}_{new} is the scaled reservoir weight matrix, $0 < \gamma < 1$ is the desired spectral radius and $\rho(W^{res})$ is the spectral radius of randomly generated reservoir Matrix W^{res}.

It is also argued that the $\rho(W^{res}) < 1$ is not a necessary condition for ESN to have the echo state property and can be achieved even when $\rho(W^{res}) > 1$ [22, 27, 21]. Intuitively, the spectral radius is a crude measure of the amount of memory the reservoir can hold, the small values meaning a short memory, and the large values a longer memory, up to the point of over-amplification when the echo state property no longer holds. The input weights 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 'tanh' activation function) whereas a very small input scaling is used wherever linearity is required in a task [27].

2.2.2 Training ESN

ESNs are mostly applied 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 is 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 [27, 21]. 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. The reservoir weight W^{res}, input weights W^{in} and W^{back} are then randomly initialized.

The training objective of ESN approach is to learn a model which outputs *y'*, such that it is as close as possible to the target output y by mining the error measure $E(y', y)$ and also generalize well on the data not used for training. Root Mean Square Error is typically chosed as erorr measure E. Thus during training, the given training input signal *u(n)* $\in \mathbb{R}^{N_u}$ and the corresponding teacher signal *y(n)* $\in \mathbb{R}^{N_y}$ is input to the reservoir at every time-step 'n'. Here n = 1,2...,T is the discrete time step for sequence of length T. The reservoir then generate a sequence *x(n)* of N_x-dimensional reservoir states which is non-linear high dimensional expansion of the input signal *u(n)* [21]. The reservoir activation and reservoir state update is computed using following recursive equations:

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

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

where *x(n)* is the vector of reservoir neuron's activations and *x'(n)* is its update at time step *n*. *tanh* is reservoir neuron activation function. $W^{in} \in \mathbb{R}^{N_x \times N_u}$ and $W^{res} \in \mathbb{R}^{N_x \times N_x}$ are input weights and reservoir weights matrices respectively. $W^{back} \in \mathbb{R}^{N_x \times N_y}$ is the optional output to reservoir matrix [3]. $\alpha \in (0, 1]$ is leaking rate of neurons.

The leaking rate, α, regulates the speed of reservoir update dynamics in discrete time. A Smaller value of α induces slow reservoir dynamics thus ensuring the long short-term memory in ESN [27, 24]. The reservoir activation states are accumulated at every time step for regression with the teacher output. The linear readout weights are then learned using equations:

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

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

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

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

$$W^{out} = YX^T(XX^T + \beta I)^{-1} \tag{2.2.6}$$

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

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

Training procedure of ESN, have 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 depends heavily on the task under consideration. Usually, a grid search is applied to explore the best parameter combination.

Chapter 3

Related Work and Open Issues

Humans have a remarkable ability of perceiving and comprehending one or more languages. But how do they know what does a sequence of symbols means? In other words, how do they link a sequence of words to its meaning? With this research question, Hinaut et al. [15] proposed a neuro-inspired model, $\theta RARes$, to process the sentence across time without having to know the semantics of the words. The model used reservoir computing approach namely echo state network and implemented on robotics architecture [16] for the thematic role assignment task. This was the first time when echo state network was used for thematic role assignment task. The experiments done for TRA showed the results toward modeling of language acquisition in brain [18, 15]. The model was based on the cue competition hypothesis of Bates et al. [5] which states that closed class words (e.g. prepositions, articles, determiners, pronouns etc.), the order of words and prosody in a sentence encodes the grammatical structure.

3.1 Overview of $\theta RARes$ Model

$\theta RARes$ Model is basically an echo state network, used to learn and predict thematic roles of the input sentences. The model is based on the notion of grammatical construction. The grammatical construction of a sentence is defined as the mapping of the surface form (word order) of the sentence to its meaning [14].Figure 3.1 represents the characterization of thematic role assignment in grammatical construction form. The model does not take in input the raw sentences but instead use the abstract form of the sentences. The abstraction marks each word of a sentence into two kinds of symbols: Function Words and Semantic Words. Semantic words are the open class words like nouns, verbs, adjectives, adverbs etc. whereas Function words are closed class words like determiners, prepositions, articles, pronouns and verb inflections like -ed,-ing or -s etc.. Thus before giving a sentence as an input to ESN, all the semantic words are replaced with a unique token 'SW' and pushed to FIFO memory stack(see fig. 3.2). The functional words were left unchanged unchanged (see table 3.1).

Thus for training the transformed sentences were presented to the model se-

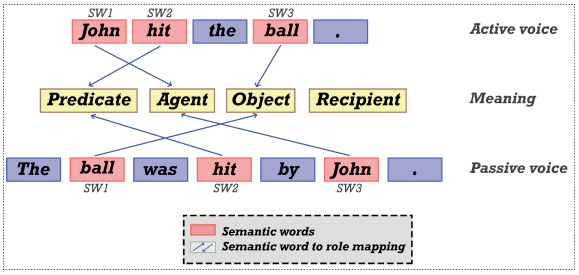

Figure 3.1: **Word2vec semantic regularities.**

Table 3.1: Transformation of a sentence by replacing semantic words with 'SW' token and localist vector representation of words used as an input for a sentence.

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

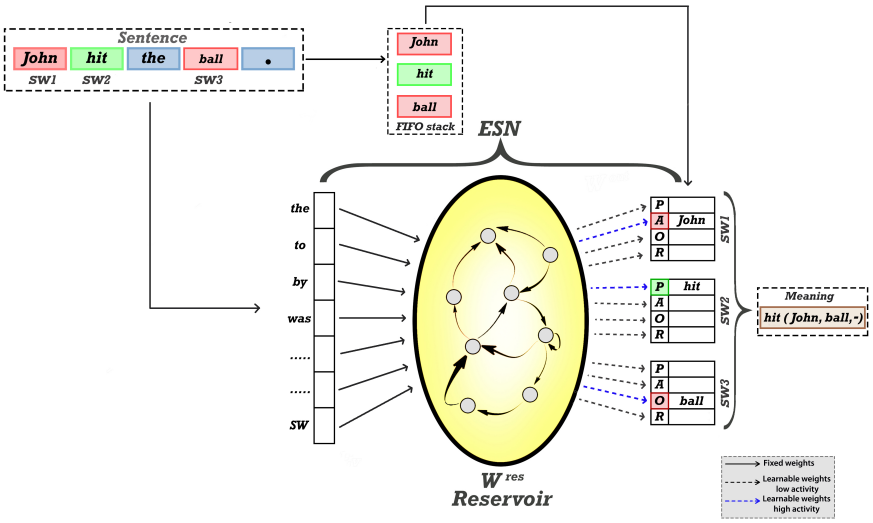

Figure 3.2: **Word2vec semantic regularities.**

quentially, word-by-word. Considering each word as a discrete atomic symbol, the words were encoded using localist vector representation, where all elements except the current input are zeros (see table 3.1). Figure 3.2 shows the functional organization of the model. The localist word vectors are projected on the input layer of ESN and coded meaning of the input sentence is teacher-forced on the output layer of ESN. The model learns the thematic roles of all semantic words by learning reservoir to readout weights of ESN. During the testing, the model predicts the coded meaning of the test sentence. The output activation is decoded to its meaning by thresholding the activation at last time step. For each SW word the that has the highest activation is considered as the role of SW. Each semantic word in the memory stack is then mapped to the corresponding role (see fig. 3.2).

3.1.1 Limitation of $\theta RARes$ model

Transforming a sentence into its abstract representation (explained above) and using the localist representation of each word in a sentence as an input to an ESN does not allow the network to leverage the information retrieved from semantically related words. 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 construction but the different coded meaning and actual meaning (see fig. 3.3). The limitation can be described below in the two ambiguous examples.

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 having the same surface form after transforming the sentence by replacing the semantic words with 'SW' token i.e. SW the SW the SW SW.

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

SW-1			
A	O	C	I
take			

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

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

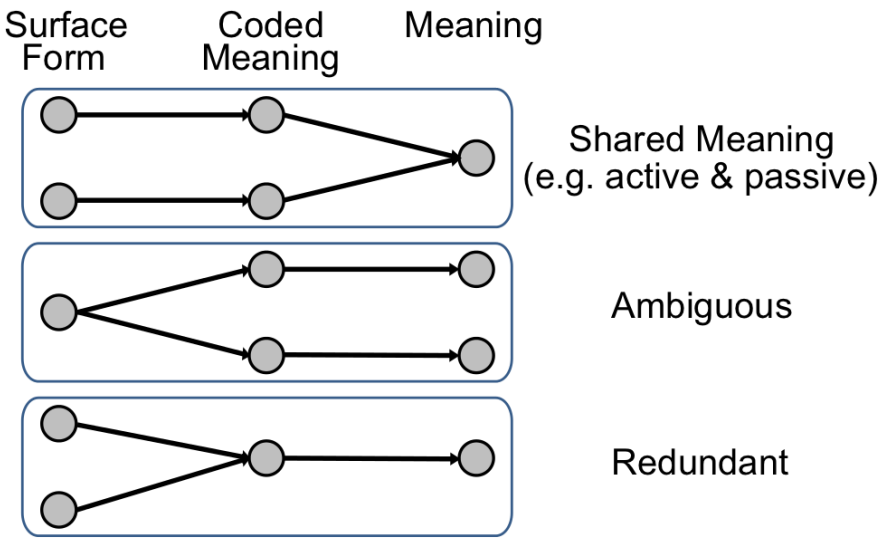

Figure 3.3: **Relation between surface form and meanings:** Sentence structure to meaning mapping exist in three categories. Active and passive sentence like "John chased the dog" and "The dog was chased by John" have the different surface form but share the same meaning i.e. chased(John, dog). Also the coded meaning of both these sentences are different i.e. SW-3 ("dog") in the active form is the 'object' whereas in passive form SW3 ("John") it is an 'agent'. Ambiguous sentences have the same surface form but different meaning and coded meaning whereas redundant sentences have different surface form but same meaning and coded meaning. Adapted from [47]

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

SW-1			
A	O	C	I
take			

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

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

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

Testing with Sentence 3: Now, when we use the sentence 3 for testing, the network suggests two pseudo probabilities for the second semantic word *'green'* i.e. being an *'Indicator'* and a *'Color'* as shown below. It becomes difficult for the model to resolve this ambiguity and assign the appropriate role. Although the word *'green'* in the test sentence 3 and the word *'blue'* in the training sentence 1 are semantically related i.e. both are colors. The model was not able to utilize this information as words were input to the in discrete forms for training. Thus by using localist vector representation we are depriving the ESN from the semantic information carried out by a words in a language.

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

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

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

Clearly we can see that the third semantic word is a *'predicate'* in Sentence (i) and *'noun'* in sentence (ii). Thus if network is trained on sentences (i) and (ii) and tested on sentence (iii) the network is not 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 model on sentence 1 Training the model on the first sentence, the fourth semantic word *'London'* is trained to be as a *location*.

Testing model on sentence 2 Testing the model with the second sentence, the fourth semantic word *learn* will be assigned the role of a location (as network was trained for this only) whereas it is actually a *predicate*. The reason for such problem is that each semantic word is considered as an unique token without taking into consideration semantics of an individual word. Hence the network could not exploit the semantic information of the words during the role assignment.

Issue with Polysemous words: In the first sentence, the second semantic word *books* is the predicate and describe the action of making 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 but they represent different meanings depending on the context. These kind of words with different meaning are also known as *polysemous* words. Semantic ambiguities because of polysemous words is hard to resolve with localist vector representation where each word is treated as an unique identifier. To resolve such kind of semantic ambiguities, distributed embedding of words can be useful as they are learned using the the contextual information of words.

3.1.2 Research Hypothesis

Use of abstract form of sentence and the localist word vectors as an input to the ESN have produced the promising results for the thematic role assignment task[15, 18]. But the behavior of the model with the distributed word embeddings was left unexplored. Thus we hypothesize that using the distributed word embeddings (e.g. Word2Vec word embeddings) could possibly resolve the problems described in the above mentioned two examples by taking into consideration the fact that they capture and encode semantic and syntactic information of words [28, 42]. Another advantage of distributed word embedding observed over localist vector representation is that the multidimensional distributed vector representation of semantically related words remains close in the neighborhood within words embedding space (see Fig. 3). This could possibly allow the model to learn these dynamics from the distributed representation of the words and avoid the disruption caused by semantically unrelated words (as in Example 1) in the sentences with the same grammatical constructions. Whereas this semantic grouping of words is not possible in the localist vector encoding as words are represented as discrete atomic symbols.

Consider using distributed embedding 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 embedding of word *'learn'* (sentence 2, Example 2) also encodes that it is a *'predicate'* along with other semantic information. When these distributed embeddings are used as an input to train the network, the learning of the model may not be disrupted by the sentences having the same surface form, but different semantic words. This is because the semantic information about the word is exposed to network and learned by it. Hence, thematic roles could be assigned to the words more accurately although the sentence has the same surface form.

Chapter 4

Approaching Word2Vec-ESN Language Model

4.1 Word2Vec-ESN Language Model

To validate our hypothesis, we propose a Word2Vec-ESN language model for TRA task. The proposed model is inspired from the $\theta RARes$ model proposed by Hinaut et al. [15] for TRA task. Word2Vec-ESN model is basically the combination of Word2Vec model and Echo State Network (ESN). Figure 4.1) shows the architecture of Word2Vec-ESN language model. The word2vec model is responsible for generating distributed embeddings of the words in the input sentences. The generated word embeddings are then used as an input to ESN, which further processes these embeddings to learn and predict the thematic roles of all semantic words in the input sentences.

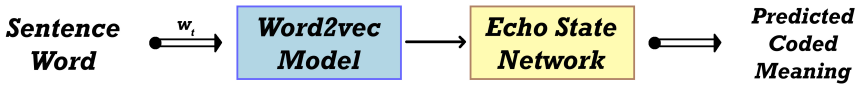

Figure 4.1: **Architecture of Word2Vec-ESN model:** The model takes the words of the input sentence as an input across time. Word2Vec model generates the distributed vector representation of the input word. The generated word vector is then used by ESN for further processing and learns to predict the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

4.1.1 Model initialization

Prior to use in the Word2Vec-ESN model the Word2Vec model was trained using skip-gram negative sampling [28] approach on a general purpose dataset (e.g. Wikipedia) and the domain specific dataset. During the training, the word2vec model learns the low dimensional distributed embeddings for each word in the corpus vocabulary (see section 5.2.1).

The reservoir in ESN, composed of leaky integrator neurons and *tanh* activation function, was sparsely and randomly initialized. A fixed fan-out of F_{hh} and F_{ih} is chosen for hidden-to-hidden and input-to-hidden connections respectively. In other words, each reservoir neuron is connected to F_{hh} other reservoir neurons and each input neuron is connected to only F_{ih} reservoir neurons. The input-to-hidden (W^{in}) and hidden-to-hidden (W^{res}) weights were generated sparse and randomly from a Gaussian distribution with mean 0 and variance 1. These weights once initialized are fixed and remains unchanged during training [20, 27].

4.1.2 Training model

Word2Vec-ESN language model treats the thematic role assignment 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 were used (see section 4.1.3). The same evaluation metrics was used for evaluating $\theta RARes$ model [15]. Thus it enable us to compare the performance of both the model.

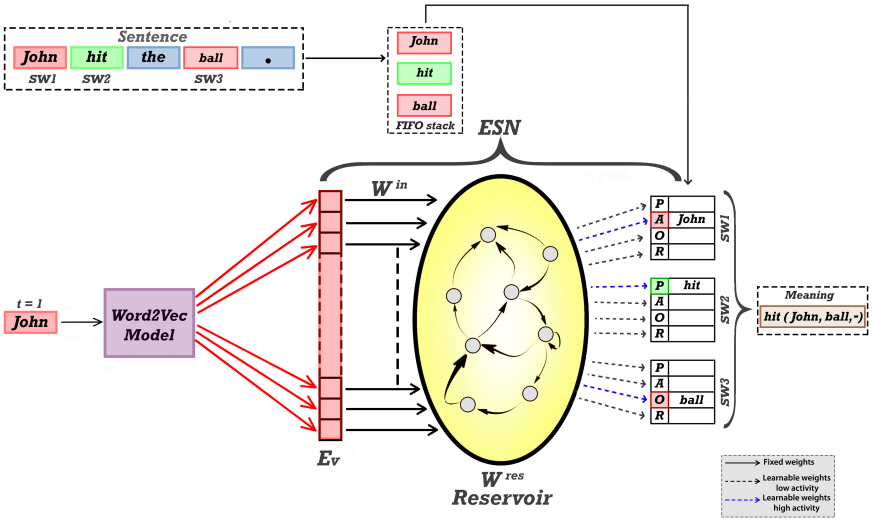

Figure 4.2: **Word2Vec-ESN language model:** The figure shows the processing of a sentence by the model variant at time step 1. Nouns and verbs (specified in red and green respectively) are stored in a memory stack for interpreting the coded meaning. The word *'John'* is input to word2vec model which generate a word vector of E_v dimensions. The output vector is then input to ESN for further processing. During training, the readout units are teacher-forced with the coded meaning of the input sentence. During testing, the readout units generate the activations which code the predicted coded meaning of input sentence. The meaning: hit(John, ball, -) is decoded from coded meaning by mapping the thematic roles with semantic words from memory stack. Adapted from [15]

Figure 4.2), shows the neural comprehension of Word2Vec-ESN model for thematic role assignment. During the training, sentences are presented to the model

one at a time, word-by-word across time. Before presenting the sentence to the model all the semantic words (e.g. nouns, verbs etc.) in the sentence are identified and placed in FIFO memory stack (see fig. 4.2). This memory stack will be used later to decode the output of the model. The readout layer of the model is also teacher-forced with the coded meaning of the input sentence. The size of readout layer thus depends on the maximum number of semantic words, any sentence can have in the corpus. A semantic word in the sentence can have one of the four possible roles: Predicate (P), Agent (A), Object (O), Recipient (R). For example, if the sentences in the corpus have a maximum of N_{sw} semantic words then the readout layer size would be $N_{sw} \times 4$ neurons; where each output neuron encodes the thematic role of a semantic word. The model could also take the topologically modified but equivalent coded meaning of the sentences (see fig4.3), where the role of each noun in a sentence is represented with respect to a verb [15]. With the topologically modified coded meaning, the readout units contains $N_{noun} \times N_{verb}$, where $N_{noun} and N_{verb}$ are number of nouns and verbs respectively. The output neurons have an activation of 1 if the corresponding thematic role is present for the sentence, -1 otherwise. The Word2Vec model receives the words from the input sentence across time and generates a word vector of E_v dimensions which is then input to the ESN. The input layer of ESN uses this word vector as input features for learning and predicting thematic roles of the sentences. Thus the size of the input layer is same as the dimensionality of word vector i.e. E_v. The output of reservoir is accumulated for each time step during the presentation of a sentence. The accumulated reservoir states are then linearly combined with readout activations to learn the reservoir-to-readout (W^{out}) weights using ridge regression. The reservoir states of ESN are reset before the presentation of the consequent sentence.

Word2Vec-ESN model can be operated in two learning modes so that it learns to extract the coded meaning of semantic words in the sentences.

1. **Sentence Continuous Learning (SCL)**: In this learning mode learning takes place from the beginning of the sentence. In other words, the coded meaning of the input sentence is made available to model from the onset of the first word of the sentence. Thus, the regression is applied with teacher roles from the onset of the first word in the sentence [15].

2. **Sentence Final Learning (SFL)**: In this mode, the learning takes places only at the end of the sentence [15]. Hence, the teacher labels are only provided to the network at the end of sentence i.e. from the last word of the sentence.

Decoding Output: As described earlier, coded meaning of a sentence is defined as the description of thematic roles for all the semantic words (e.g. nouns, verbs etc.) in the sentence. As the readout neurons of model codes the thematic roles for individual semantic words, the output activations produced by the model during testing for a sentence are thresholded at 0 for every semantic word and then the

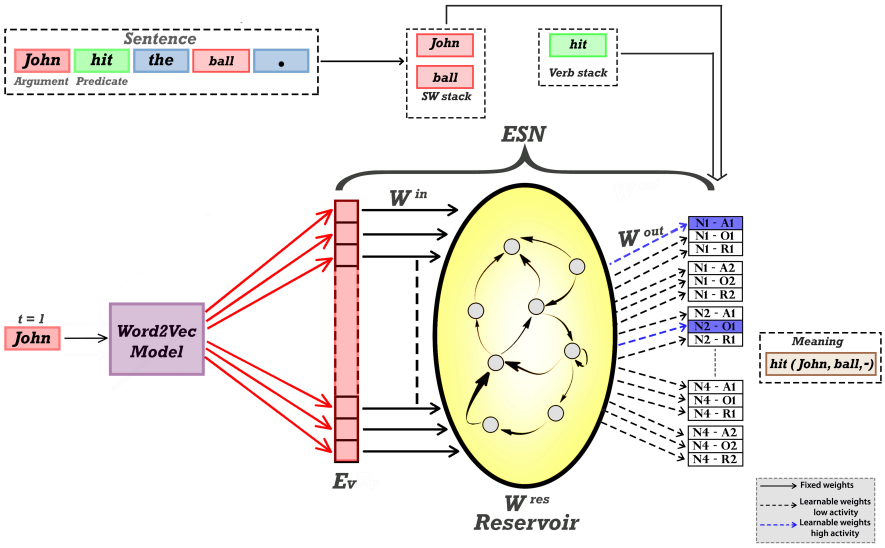

Figure 4.3: **Word2Vec-ESN language model with toplogically modified output coding:** The figure shows the processing of a sentence by the model variant at time step 1. Nouns and verbs (specified in red and green respectively) are stored in a repective memory stack for interpreting the coded meaning later. The word *'John'* is input to word2vec model which generate a word vector of E_v dimensions. The output word vector is then input to ESN for further processing. During training, the readout units are teacher-forced with the coded meaning of the input sentence. The coded meaning "N1-A1" represents that Noun-1 ('John') is agent of Verb-1 ('hit'). During testing, the readout units generate the activations which code the predicted coded meaning of input sentence. The meaning: hit(John, ball, -) is decoded from coded meaning by mapping the thematic roles with nouns and verbs from memory stack. Adapted from [15]

maximum of all activation between 4 possible roles (i.e. Predicate, Agent, Object, Recipient) is taken as the coded meaning of a semantic word [15]. A semantic word is said to have an incorrect coded meaning if the winning readout neuron is not the correct 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 [15]. The coded meaning of the sentence can then be decoded to the actual meaning by mapping the coded meaning to the semantic words from the FIFO memory stack(see fig 4.2).

4.1.3 Evaluation Metrics

To evaluate the performance of the model we used the same two error measures that were used by Hianut et al.[15] for TRA task: the Meaning Error (ME) and the Sentence Error (SE). Meaning error is the percentage of semantic words with incorrect coded meaning, whereas the sentence error is the percentage of sentences having at least one semantic word with incorrect coded meaning. Both the error

measure are related but there is no strict corelation between them [15]. Sentence error is a more stricter measure than meaning error to evaluate model performance because a meaning error of 5% cannot be used to estimate the sentence error, as these 5% incorrect words can be from just one sentence or several sentences.

To evaluate the model not all the readout neurons are considered i.e. if a sentence have only 3 semantic words then only readout neurons corresponding to these two semantic words were analyzed [15] and remaining coded meaning of remaining neurons are ignored. If there are more than one verb, in a sentence then each semantic word can have possible role with respect to each verb.

4.2 Variant of Word2Vec-ESN Model

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

Two input features play an important role in this model variant: argument and predicate, with an argument describing the current word being processed and predicate, describes the verb with respect to which argument is processed. So, if there are N_v verbs in a sentence then the same sentence is processed N_v times. Each argument then takes a unique role for an argument-predicate pair. For example in the following sentence there are two predicates namely *'chased'* and *'ate'*. Thus this sentence will be processed twice and each argument will take a role for an argument-predicate pair [47].

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

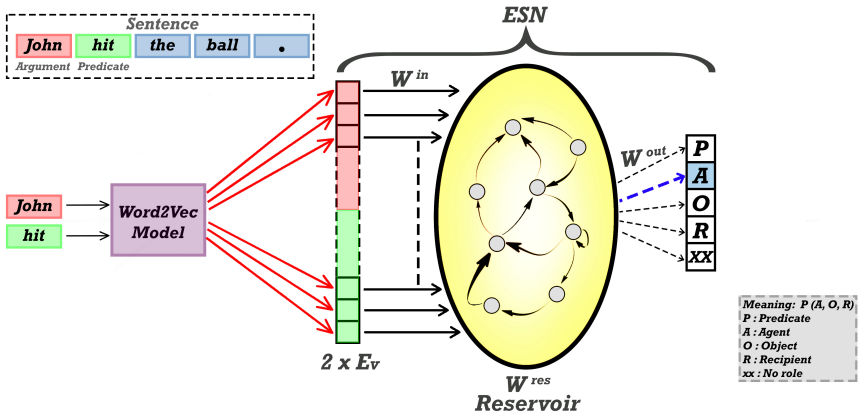

Figure 4.4: **Word2Vec-ESN Model Variant:** The figure shows the process of a sentence in model variant at time step 1. At any instant of time, an argument (current word, marked in orange) and predicate (marked in green) is input to the model. Word2Vec model generates the word vectors of E_v dimensions which are then concatenated to form a $2 \times E_v$ dimensions(shown in orange and green color). ESN takes the resultant vector for further processing. During learning, the readout neurons are presented with the role of input word (e.g. 'A' for Agent). The readout weights (shown in dashed line) are learned during training. During testing the readout unit codes the role of semantic words, which are then accumulated and decoded to form the meaning *hit(John,ball,–)* of the input sentence at the end. Inspired from [15]

4.2.1 Training model variant

To train the model variant, training sentences are presented to the model sequentially. An input sentence is processed as many time as there are verbs in the sentence, forming multiple sequences. Thus the model takes an argument-predicate pair across the time as an input. The readout layer has 5 neurons each coding for a role (P, A, O, R, XX). Thus during training, the role of the input argument-predicate pair is also teacher-forced to the model. An output neuron have an activation 1 if the input argument-predicate pair has the corresponding role, -1 otherwise. The Word2vec model initially receives an argument-predicate pair of the input sentence and generates the distributed embeddings for both the input words. The generated word embeddings are concatenated and then taken by ESN as an input. Thus the size of ESN input layer is $2 \times E_v$ where the first E_v neurons takes the vector representation of the argument and remaining E_v neurons for the predicate. The reservoir internal states are collected for an input sequence over time which will be used later for regression with the desired output. Reservoir-to-readout (W^{out}) weights are then learned by using ridge regression over collected reservoir states and the readout activations. Figure 4.4 shows the processing of an example sentence by this model variant.

4.2.2 Decoding Output

Recall that the model variant process a sentence as many times as there are verbs in the sentence. Thus during testing, the output activation of the model is used to predict the role for an argument-predicate pair. The role having highest output activation is considered as the role of an argument-predicate pair [2]. The role for all argument-predicate pairs is collected and the meaning of the sentence with respect to a verb can then be interpreted by filling up the tagged words in P(A, O, R). For example in the sentence "John hit the ball." as shown in figure 4.4, the roles for each word (i.e. A P XX O) is used to get the meaning of the sentence as "hit(John,ball,–)".

4.2.3 Evaluation Metrics

To analyze the performance of this model variant on thematic role assignment task, the confusion matrix or contingency table [1] was used and classification scores: Accuracy, Precision, Recall, and F1-Score were calculated for all possible roles. The classification scores were then macro-averaged, to get a 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 [32]. Higher the classification scores the better. The same evaluation metrics was also used for CoNLL-04 and CoNLL-05 semantic role labeling shared task [8, 7].

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

The confusion matrix describes the predictions made by the model. The rows of the matrix correspond to the actual roles and the columns correspond the predictions made by the model. The diagonal elements of this matrix represent the number of words for which the predicted role is equal to the true role, whereas all the non-diagonal elements represent the number of words which were labeled incorrectly. As the values of diagonal elements of the confusion matrix indicates the number of correct predictions so higher the values of diagonal elements the better.

Using the confusion matrix accuracy can be calculated as the ratio of 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 [39].

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

However, accuracy measure can be distorting (because of accuracy paradox) when the dataset has words with large role imbalance as it gives high scores to models which just predict the most frequent class and cannot be used alone to evaluate the model performance [41, 45]. In our dataset, we have imbalanced roles as most of the words are labeled as "XX" (No Role) compared to other roles (P, A, O, R). Thus we needed additional measures such as Precision, Recall, and F1-score to evaluate the model. All these scores are reported as a value between 0 and 1.

Precision is defined as ratio of True positive (TP) to False Positive(FP) and True Positive (equation 4.2.4). It is the measure of the accuracy of a role provided that a specific role has been predicted [39].

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

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

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

Recall is defined as the ratio of True Positive to True Positive and False Negative. It measures how good the model is in labeling the correct roles. It is also called *'Sensitivity'* or *'True Positive Rate'*. [39].

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

Recall for the role 'A', from the above confusion matrix will be:

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

F1-Score or (F1) is the harmonic mean of precision and recall. In other words, it represents the balance between the precision and recall. It takes false positive and false negative in account. This score is really useful whenever there is class imbalance in the dataset [39] and is calculated as:

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

Chapter 5

Experiments

This chapter presents the experimental performed in this Dissertation. The rest of the chapter is organized as follows. First, we will have an overview of the corpora used to perform the experiments. Then we describe the experimental setup used to perform the thematic role assignment task with the Word2Vec-ESN model and the variant proposed in the previous chapter. We also describe the experiments performed for thematic role assignment task.

5.1 Corpora and pre-processing

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

5.1.1 Corpora For TRA Task

In order to have a fair comparison between the Word2Vec-ESN model and $\theta RARes$ model (described in section 3.1), we used the same coprora[1] used by Hinaut et al. [15, 16] to perform TRA task with $\theta RARes$ model. Thus, the corpus-373, corpus-462, corpora-90582 containing 373, 462 and 90582 sentences respectively were used to perform the Experiments with the proposed model for TRA task.

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

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

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

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

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

Corpus-462 and Corpus-90582: The sentences in corpus-462 and corpus-90582 were generated by Hinaut et al. using a context-free-grammer for English language and used for TRA task [15]. Each sentence in these corpora have verbs which takes 1, 2, 3 clause elements. For example, the sentences, 'The man *jump*', 'The boy *cut* an Apple', 'John *gave* the ball to Marie', have verbs with clause elements agent, agent and object, or agent, object and recipient respectively. The sentences in the corpora have a maximum of four nouns and two verbs [15]. A maximum of 1 relative clause is present in the sentences; verb in the relative clauses could take 1 or 2 clause elements (i.e., without recipient). For e.g. 'The dog that bit the cat chased the boy'. Both the corpus-462 and corpus-90582 have the constructions in form:

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

Each construction in the corpus was divided into four parts. The first part describes the meaning of sentence using semantic words (or open class words) in order of predicate, agent, object, recipient. The second part (between '$<o>$' and '$</o>$') describes the order of thematic roles of semantic words as they appear in the raw sentence. The third part (between ';' and '#') contains the raw sentence with verb inflections (i.e. '-s') and the fourth part is the abstract representation of a sentence with semantic words removed and replace with 'X' [15].

Corpus-90582 have 90582 sentences along with the coded meaning of each sentence. This corpus is redundant; multiple sentences with different grammatical structure but the same coded meaning (see fig. 3.3). In total, there were only 2043 distinct coded meanings [15]. This corpus also has an additional property that along with complete coded meanings for sentences it also have incomplete meanings. For example, the sentence "The Ball was given to the Man have no *'Agent'*, and thus the meaning of the sentence is "given(-, ball, man)". The corpus also contains 5268 pair and 184 triplets of ambiguous sentences i.e., 10536 and 553 sentences respectively. Thus in total there were 12.24% (i.e., $5268\times2+184\times3 = 11088$) of ambiguous sentences which have the similar grammatical structure but different coded meaning [15].

Corpus-373: Apart from the corpus-462 and corpus-90582 which were artificially constructed using the context-free grammar, we also used the corpus-373. Corpus-373 includes the instructions collected from the participants interacting

with a humanoid robot (iCub) in a Human-Robot Interaction study of language acquisition conducted by Hinaut et al. [16]. In the study, the robot first performs one or two actions by placing the available objects to a location (e.g. left, right, middle) and the participants observes the actions. Then the participants were asked to instruct the robot again to perform the same actions in natural language. Thus the corpus contains 373 complex instructions to perform single or double actions with temporal correlation (see Action Performing task in Hinaut et al. [**?**] for more details). For example, the instruction "Point to the guitar" is a one action command whereas the instruction "Before you put the guitar on the left put the trumpet on the right" is a complex command with double actions, where the second action is specified before the first action. A list of 86 closed class words are also provided with this corpus. Also, unlike corpus-462 and corpus-90582 this corpus does not contain verbs inflections. Thus, this data is complex enough to test the learnability and generalization of the proposed model on TRA task.

Pre-processing: We modified the sentences in this corpus-45 by replacing the token 'N' and 'V' with appropriate nouns and verbs such that the same 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 have sentences where are verbs are represented along with inflections (suffixes "-s", "-ed", "-ing"). We preprocessed these constructions in corpus-462 and corpus-90582, to obtain the raw sentences without verb inflections. Firstly, all the words are lowercased and then the verbs with inflections are replaced by conjugated verbs [2]. The verb conjugation to be used depends on the inflection used for the verb. For example, the sentences "The giraffe walk -s" and "John eat -ed the apple" has been changed to "The giraffe walks" and "John ate the apple" respectively. This preprocessing was done because the distributed word representation captured by word2vec model already captures these syntactic relations which was imposed previously using verb inflection e.g $vector('walks') - vector('walk') \approx vector('talks') - vector('talk')$.

5.1.2 Corpus For Training Word2Vec Model

In this work, To train the word2vec model, we used the Wikipedia corpus[3] ($\approx$ 14 GB) to obtain the low dimensional distributed embeddings of words. The corpus contains 2,65,8629 unique words. We chose to use Wikipedia data because we needed a general purpose dataset to get the vector representation of words. The Word2Vec model does not give good quality vector representation for words when trained over a small corpus thus a general purpose data set with billions of words is required to have good word embeddings. Thus more words we have the better the vector representation of words. Once the vector representation of words in Wikipedia data is obtained, the model can then be trained further on any

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

our domain specific dataset (corpus-462 and corpus-90582) with more bias toward domain specific dataset (by repetition of dataset during training) to update the previously learned vector representations.

5.2 Experimental Setup

5.2.1 Obtaining Word Embeddings

Before using the model for TRA task, we need to have the word embeddings. So in order to get the vector representation of words we first trained the Word2Vec model on Wikipedia dataset. For training, the word2vec model with skip-gram negative sampling (see Chapter 2 for more details) approach was used to obtain the word embeddings as it was claimed to accelarate the training and generates better word vectors which performs better on word analogy task[29, 28].

To obtain the word vectors of different dimensions, 6 word2vec models were trained with the hidden layer containing 20, 30, 50, 100, 200, 300 units respectively. Thus each model produce a distributed word embedding corresponding to the size of hidden layer. All the 6 models were trained using same hypermeters. A context window of ± 5 was used. The negative sampling size of 5 was chosen i.e. 5 noise words are chosen randomly from the vocabulary which does not appear in the context of the current input word. As the Wikipedia corpus is huge, it contains some words like "a", "the", "in" etc. which occurs million of times. Thus the frequent words are discarded with the probabily using equation 2.1.10 with a subsampling threshold of $t = 10^{-5}$. We ignored all the words which appear less than 5 times in the corpus. To update the network weights stochastic gradient descent was used [38, 28]. The initial learning rate was set to be $\alpha = 0.025$, which drops to $min_alpha = 0.0001$ linearly as training progress.

The word embeddings obtained from training on Wikipedia dataset are accurate enough to capture the semantic relationship of words for e.g. $vec(Paris) - vec(France) + vec(Germany) \approx vec(Berlin)$. Before training the model on Wikipedia corpus, a vocabulary of words is created and once the vocabulary is created it is not possible to add new words to this vocabulary. However, there is a possibility that when a domain specific corpus (e.g. corpus-462, corpus-373 etc.) is used to further train the Word2Vec model, some of the new words may not be present in previously generated vocabulary. Due to this limitation, it was not possible to get the distributed embeddings of these new words. Thus we needed to update the vocabulary of the model, if the new words are not present in the vocabulary in order to facilitate the online training of Word2Vec model. Unfortunately, neither C++ API [4] nor Gensim python API [37] implementation of Word2Vec supports vocabulary update, once created. So, we implemented the online training[5] of word2vec by modifying and extending the Gensim API. The new words that were not present in the existing vocabulary is now added and initialized with some random weights.

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

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

The model can then be trained in the usual manner to learn the distributed embeddings of new words. Although now the vocabulary can now be updated in an online manner but the vector embedding of the newly added words has poor quality if its count in the new corpus is less. This can be improved by repetition of new dataset several times before training the model [6].

So now when we have an online version of training word2vec model, we extend the word2vec model by resuming the training on the domain specific corpus (corpus-462 and corpus-90582). While updating the model on the new dataset we do not disregard any word irrespective of its count, so that we have vector embeddings of all the words in our domain specific corpora. Once the Word2Vec model is trained, the generated word embeddings was normalized using L-2 norm before using them. On important point to remember while training word2vec model is that if a already trained model has to be trained further on any other corpus, then normalization should not be done, as it is not possible to train the model again with normalized word vectors [37]. The trained word2vec model is now ready to be used with Word2Vec-ESN model.

5.2.2 ESN reservoir initialization

The size of the reservoir is one of the important parameters of ESN and is often recommended to use a bigger reservoir that can be computationally afforded provided the appropriate regularization method is used [27]. The bigger the size of the reservoir, the easier it is for the ESN to learn from the input signal. Thus we chose a reservoir of 1000 (until and unless specified) leaky integrator neurons with *tanh* activation function. The input-to-hidden, hidden-to-hidden weights were randomly and sparsely generated using a normal distribution with mean 0 and variance 1. Thematic role assignment does not use any feedback from the readout layer, we do not use any output-to-hidden weights. The state update equations 2.2.2 and 2.2.3 thus changes to:

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

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

To generate the sparse weights a fixed fanout number of $F_{hh} = 10$ and $F_{ih} = 2$ was used i.e. each reservoir neuron was randomly connected to 10 other reservoir neurons and each input neuron was connected to only 2 other reservoir neurons. Use of fixed fanout number scales the cost of state update of reservoir linearly with increase in reservoir size [27]. After weights initialization, there are several parameters which are to be optimized which are task specific. In the next subsection we describe the optimization of reservoir parameters.

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

5.2.3 Learning ESN parameters

Echo state network have several parameters to be optimized for the proposed model to perform efficiently on thematic role assignment task. Some of the parameters like reservoir size, sparsity, distribution of non-zero weights are straightforward [27]. Whereas other parameters like Spectral Radius (SR), Input Scaling (IS) and Leak Rate (LR) are task dependent and are often tuned by multiple trials. Thus to identify these parameters, we performed a grid search over the parameter space using 5 reservoir instances. As the parameter search space can be really large, a broader grid with wider parameter ranges was first explored to find the optimal grid: grid with low sentence error for Word2Vec-ESN model and high F1-Score for the model variant. The optimal region identified during the broder grid search was then used for the narrow search to identify the optimal parameters [27]. As both the proposed model and its variant process the sentences differently and have different training objectives, the ESN parameters for both the model and its variant were optimized separately. Also, the Word2Vec-ESN model parameters are optimized separately for sentence continuous learning mode and in sentence final learning mode. Here we describe about parameter optimization for the general case for corpus-462.

Word2Vec-ESN model: To optimize the reservoir parameters, corpus-462 was used. To get the optimal paramaters (i.e. parameters producing the lowest sentence error) of Word2Vec-ESN model, the model was trained and tested over a range of parameters using 10-fold cross-validation method; corpus-462 with 462 sentences was randomly split into 10 equally sized subsets (i.e. each subset with $\approx$ 46 sentences). The model was trained on sentences from 9 subsets and then tested on remaining one subset. This process was repeated 10 times such that the model was trained and tested on all the subsets at least once. A reservoir of 1000 neurons and a fixed regularization coefficient, $\beta = 1e{-}6$ for ridge regression was used. By exploring the parameter space we identified the optimal parameters as $SR = 2.4$, $IS = 2.5$ and $LR = 0.07$ in SCL mode and $SR = 2.2$, $IS = 2.3$ and $LR = 0.13$ in SFL mode.

Word2Vec-ESN model variant: For the Word2Vec-ESN model variant, we find the optimal parameters (i.e. parameters producing highest F1-Score) during the grid search using the same corpus-462 and by applying 10-fold cross-validation. Keeping all the parameters i.e. reservoir size and regularization coefficient, identical to Word2Vec-ESN model described above, optimal parameters were identified as $SR = 0.7$, $IS = 1.15$, $LR = 0.1$. As later in the experiment 2, we will also be testing the performance of the model variant on the sentences transformed to grammatical form (i.e. semantic words replaced with 'SW' token) and using localist word vectors. We also need to find the optimal parameters for this transformed dataset. Keeping all the conditions same, we performed a grid search over parameter space and identified the optimal parameters as $SR = 1.3$, $IS = 1.0$, $LR = 0.4$.

5.2.4 Input and Ouput Coding

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

Word2Vec-ESN model: A raw sentence is presented to the model, where each word in the sentence is processed across time by both word2vec model and ESN. The word2vec model outputs the $E_v = 50$ dimension word embedding which is then used as an input for ESN. Thus input layer of ESN has 50 neurons.

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

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 have one of the 3 possible thematic roles (Object, Predicate, Location) with respect to maximum of two actions [16].

Output neuron has an activation 1 if the role is present in the sentence, -1 otherwise. When using corpus-90582 for training the number of neurons in the reservoir were raised to 5000 and also the readout neurons are increased to 30 ($5 \times 3 \times 2$) as there was maximum of 5 nouns in the sentences of this corpus.

Word2Vec-ESN model variant: Recall that in Word2Vec-ESN model variant 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). A sentence is processed as many time as there are verbs in the sentence. The word2vec model firstly 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 input layer thus has 100 neurons where first 50 neurons encodes the vector representation of the word and remaining 50 neurons codes for the verb with respect to which word is being processed. Unlike the model variant-1, the size of readout neurons always remains the same and contains 5 neurons each coding for a role: Predicate (P), Agent (A), Object(O), Recipient (R) and No Role (XX) for both corpus-462 and corpus-90582. Readout neuron of ESN has an activation 1 if the input word-verb (argument-predicate) pair have the corresponding role, -1 otherwise.

5.3 Experiments

In this section we descibe the experiment performed in this thesis. Untill and unless specified we use all the optimal parameters identified during the grid search

for both Word2Vec-ESN model and its variant, described in section 5.2.3.

5.3.1 Experiment-1: Model performance on small corpous

In order to determine the model's capability for predicting thematic roles of the sentences using word2vec embeddings for words, we first used the limited set of sentences i.e. 26 sentences (sentence 15 to 40) from corpus-45. The chosen sentences have distinct surface form and grammatical structure (e.g. active, passive, dative-passive). Using these limited set of sentences we performed the simulations both on Word2Vec-ESN model and its variant. Both the models were first trained and tested on all the sentences to get the training errors and then tested using Leave one Out (LoO) cross-validation method: training on 25 sentences and testing on 1 sentence, such that all the sentences are tested atleast once.

For simulation with word2Vec-ESN model a reservoir of 1000 neurons and reservoir parameters, SR=? , IS=?, LR=? was used. Whereas for simulations with model variant a reservoir of 500 neurons with reservoir parameters SR=? , IS=? , LR=?, was used. The parameters for both the model variants were found by exploring the parameter space using LoO cross-validation. This experiments remains a toy demonstration and we will explore the generalization capabilty of the model further with the an extend corpus in experiment 2 and 5.3.5.

5.3.2 Experiment-2: Generalization Capabilities

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

Word2Vec-ESN model: We performed simulations on Word2Vec-ESN model using 10-fold cross validation. A reservoir of size 1000 neurons was used. For this experiment we used the equivalent topologically modified output coding. In order to compare the generalization ability of the Word2Vec-ESN model and $\theta RARes$ model, simulations were performed in both the learning modes i.e. in SCL and SFL mode. The reservoir parameters identified during grid search for both the learning modes were used i.e. $SR = 2.4$, $IS = 2.5$, and $LR = 0.07$ in SCL mode and $SR = 2.2$, $IS = 2.3$ and $LR = 0.13$ SFL mode (see section 5.2.3).

Word2Vec-ESN model variant: Recall that the Word2Vec-ESN model variant process the sentences diffently from Word2Vec-ESN model (see section 4.2 in chapter 4). Thus in order to compare the effect of processing the raw sentences with distributed word embeddings over the transformed sentence in grammatical form with localist word vectors, simulations were performed on the model variant in two different configurations. Simulations in both the configurations were performed using a 5 reservoir instances of 1000 neurons.

1. **Configuration-1:** In this configuration, the model variant process the sentences using the standard functioning of model variant i.e. raw sentences were used and word2vec embeddings generated by Word2Vec unit are input to ESN (see section 4.2 for more detail). For performing simulations in this configuration, the reservoir parameters $SR = 0.7$, $IS = 1.15$, $LR = 0.1$ were used. The parameters are indentified by exploring the parameter space.

2. **Configuration-2:** In this configuration, the model variant is used without word2vec unit and only ESN is used for processing the sentence. However, all the functioning remains the same. The input sentences were transformed in grammatical form: all the semantic words in the sentence were replace with 'SW' token, and localist word vectors are input to ESN. For simulation the reservoir parameters $SR = 1.3$, $IS = 1.0$, $LR = 0.4$ found previously during grid search were used.

5.3.3 Experiment-3: Effect of Corpus structure

As described previously, that the sentences in the corpus-462 were created based on context-free grammar. Thus the sentences in the corpus contain an inherent grammatical structure. The models thus possibly utilizes the underlying grammatical structure to some extent for learning and generalizing. To test this hypothesis and to demonstrate that the model is not generalizing on any other inconsistent regularity in the corpus, in this experiment, the inherent grammatical structure from the sentences was removed by randomizing the word orders within the sentences [15]. 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. The situation of overfitting typically occurs when the corpus size is significantly less than the number of trainable parameters [15]. In our case with corpus-462, the number of trainable parameters is significantly greater than our corpus size (i.e. 462 sentences). This is thus a possible situation of overfitting.

In this experiment, the corpus-462 with the scrambled sentences (i.e. in absence of any grammatical structure) was presented to Word2Vec-ESN model. Keeping all the conditions and the reservoir parameters same as used in Experiment-2, a 10 fold cross-validation was performed with 5 model instances. The cross-validation errors obtained in the Experiment-2 on the corpus-462 with inherent grammatical structure can then be compared with the cross-validation error obtained while using scrambled corpus. If the model is not overfitting and learning from the grammatical structure then the model should not perform better on the corpus with scrambled sentences (i.e. in absence of grammatical structure). However, in the case of overfitting the generalization effect on the corpus should not vary much both in presence and absence of grammatical structure [15].

5.3.4 Experiment-4: Effect of Reservoir size

The most important hyperparameter which effects the performance of the model is the size of the reservoir (i.e. number of neurons in the reservoir). Addition of neurons in the reservoir is also computationally inexpensive because the read-out weights (W^{out}) scales linearly with the number of neurons in the reservoir [43]. So, in order to determine the effect of reservoir size on the performance of the Word2Vec-ESN model and its variant, the simulations were performed over a range of reservoir size [1] [2] using 5 instances of the model i.e. reservoir weights of the model were initialized using 5 different random generator seeds).

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

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

5.3.5 Experiment-5: Effect of Corpus size

In order to investigate the effect of corpus size and determine the scaling capability of the model, we used extended corpus-90582 (see section 5.1) for this experiment. As this corpus also contains 12% of ambiguous sentences which impedes the learning and generalization of the model, this experiment will also validate the model's ability to process the abmbiguous sentences. Thus, in order to study the scaling capabilty of the model with different corpus size, 6 sub-corpora were created by randomly sampling 6%, 12%, 25%, 50%, 75%, 100% of sentences from the orginal corpus of 90582 sentences [15].

Word2Vec-ESN model: Each of the generated sub-corpora was exposed to the model and a 2-fold cross validation was performed where 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 only performed with Word2Vec-ESN model in both the learning modes i.e. SCL and SFL mode, using a 5 model instances each with a reservoir of 5000 neurons. All other parameter were kept identical to the Experiment-2.

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

[2] Reservoir sizes explored in model variant: [50, 100, 250, 400, 600, 800, 1050, 1600, 2250, 3320, 3860, 4500]

5.3.6 Experiment-6: Generalization on new corpus

One may argue that the previously used corpus (corpus-462 and corpus-90k), which were artifically constructed using grammer is adding a bias to the model which can may make it easier for the model to learn and generalize on these corpora for thematic role assignment task. To answer this question, in this experiment we used the corpus-373 (see section 5.1) collected in a Human Robot Interaction (HRI) study of language acquisition.

Word2Vec-ESN model: To test the generalization of Word2Vec-ESN model on corpus-373, simulations were performed using 10 model instances, in SCL mode and with leave-one-out (LoO) cross validation method. We chose SCL mode and LoO cross-validation method, so that results can be compared with that of obtained using $\theta RARes$ model [16]. Also, to make the results of both the model deterministically comparable, topologically modified coding was not used in this experiment. For this experiment, reservoir parameter space was not explored to identify optimal reservoir parameters, but instead, used the optimized parameters obtained on corpus-462 and used in Experiment-2 i.e. $SR = 2.4$, $IS = 2.5$ and $LR = 0.07$. Doing so will enable us to test the robustness of the model parameters, learned previously using corpus-462 with topological modified coded meaning, on the new corpora.

5.3.7 Experiment-7: Effect of Word2Vec word dimensions

In all the previous experiments, we used the word vectors of 50 dimensions. However, Mikolov et al. [29, 28] showed that the word vectors of higher dimensions (e.g. 300 in their case) performs better on word analogy task. Thus in this experiment, the effect of different dimensional word vectors on TRA task will be explored. This experiment was only performed on Word2Vec-ESN language model and not on its variant.

Word2Vec-ESN model: Recall that in section 5.2.1, 6 Word2Vec model were trained each to generate the word vectors of 20, 30, 50, 100, 200, 300 dimensions respectively. Thus, each of these models were used in Word2Vec-ESN language model to test for the effect of respective word vector dimensions. For this experiment, the corpus-373 without the topologically modified coded meaning was used (see fig. 4.2). Simulations were performed with 10 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. All the reservoir parameters were kept identical to Experiment-2.

Chapter 6

Results and Discussion

6.1 Results and Discussion

6.1.1 Experiment-1: Model performance on a small corpous

Word2Vec-ESN Model:

Model Variant-2: The model Variant-2, on the other hand, with a reservoir of size 600 neurons, produced the classification scores [write score here] during cross-validation.

6.1.2 Experiment-2: Generalization Capabilities

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

We compared the performance of Word2Vec-ESN model with $\theta RARes$ model which takes the sentences in grammatical form and words are represented in localist fashion. As illustrated in Table 6.1, one can see that $\theta RARes$ model, in SFL mode, learned all the constructions thouroughly during training whereas the Word2Vec-ESN model learned with small errors. During testing, we can see an improvement of 8.04% sentence error in SCL mode with Word2Vec-ESN langugae 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-ESN and $\theta RARes$ model. In SFL mode, both the meaning and sentence errors remained nearly same in both the models. It can also be observed that with Word2Vec-ESN model, the performance of the model is improved mainly in SCL mode as compared to SFL mode, whereas it was vice-versa in $\theta RARes$ model.

Table 6.1: Mean and standard deviation of meaning and sentence error on train and test set of coprus-462 in different learning modes.

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

Meaning (ME) and Sentence error (SE) in different learning modes with Word2Vec-ESN model using distributed word embeddings and *θRARes* model [15] which uses grammatical form and localist representation of words of sentences. The errors are given in percentage with 2 decimal precision. SCL: Sentence Continuous Learning; SFL: Sentence Final Learning; std: standard deviations. Simulations were done with 5 model instances with reservoir of 1000 neurons.

Word2Vec-ESN model variant: Table 6.2, illustrates the training and cross-validation classification scores of simulations performed with the model variant in both the configurations as described in Experiment-2 (see section 6.1.2). The word2vec-ESN model variant, in configuration-1, when trained and tested on all 462 sentences of corpus-462, the model learned to label the argument-predicate pairs in the sentences with Precision (Pr), Recall (Re) and F1-Score (F1) of 97.46%, 92.29%, 94.65% respectively. When tested using 10-fold cross validation for generalization, the model genralized with Pr = 96.76%(±0.08%), Re = 91.78%(±0.08%) and F1 = 93.99%(±0.09%). Notice the marginal difference between the training and cross validation scores, indicating that the model variant in configuration-1 is generalizing well on untrained sentences and is not overfitting.

When using the sentences transformed to GF and localist word representations as an input to ESN (configuration-2), the model variant learned the sentences with Pr = 61.96%(±0.07%), Re = 67.69%(±0.29%), F1 = 63.55%(±0.17%) during training. With 10-fold cross-validation the model generalized with Pr = 61.91%(±0.07%), Re = 68.20%(±0.46%) and F1 = 63.71%(±0.22%). It can be noticed that the model variant in this configuration was not able to learn the thematic role during training and hence not able to generalize during testing.

Figure 6.1 shows the confusion matrix, plotted using cross-validation results of the model variant in configuration-1 and configuration-2 (See section 6.1.2).

Table 6.2: Classification scores produced by Word2Vec-ESN model variant on corpus-462 in two configurations.

		Configuration-1	Configuration-2
Precision	test	96.76 (± 0.08)	61.91 (± 0.07)
	train	97.46 (± 0)	61.96 (± 0.07)
Recall	test	91.78 (± 0.08)	68.20 (± 0.46)
	train	92.29 (± 0.22)	67.69 (± 0.29)
F1-Score	test	93.99 (± 0.09)	63.71 (± 0.22)
	train	94.65 (± 0)	63.55 (± 0.17)

Classification scores (in %) of Word2Vec-ESN model variant during training and testing condition. Configuration-1: word2vec word vectors are input to ESN. Configuration-2: Sentences transformed to grammatical form and localist word vectors are input to ESN. Simulations were done with 5 instance of model variant with reservoir of 1000 neurons.

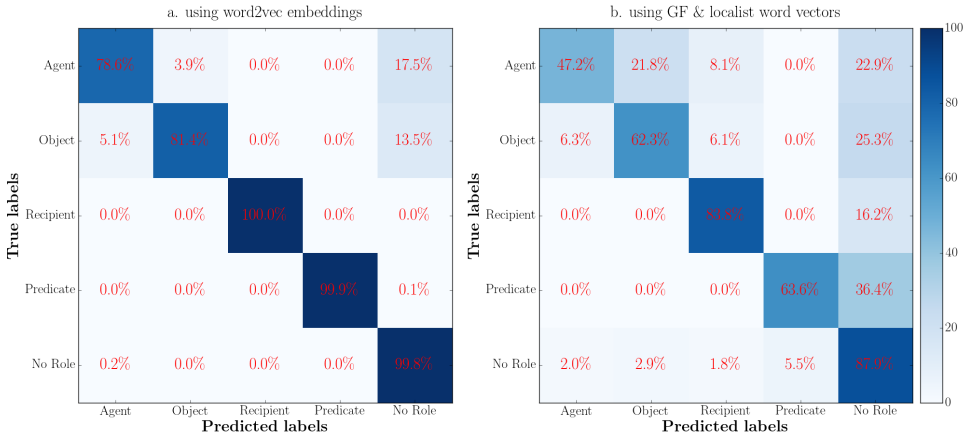

Figure 6.1: **Normalized confusion matrix with Word2Vec-ESN model variant:** The confusion matrix with true roles (in rows) and predicted roles (in columns). The top-left to bottom-right diagonal shows the percentage of words whose roles are predicted correctly. Everything other than this diagonal represents the incorrect prediction of roles. Model identified almost all words labelled as Recipient , Predicate and No Role and made some errors in predicting role Agent and Object. The results were obtained with reservoir of 1000 neurons and 10 fold-cross validation.

The corresponding classification scores of individual roles produced by model variant during 10-fold cross-validation in both the configuration are also reported in Table6.3.

As seen in confusion matrix (see fig. 6.1), when using word2vec word vectors as an input to ESN, the model predicted the words with role 'Recipient', 'Predicate' and 'No Roles' almost without making any errors. The model predicted 78.6% of the words with role 'Agent' correctly, whereas 3.9% and 17.5% of the words with role 'Agent' were incorrectly predicted as 'Object' and 'No Role' respectively. Similarly, 81.4% of the words with role 'Object' were correctly classified, whereas 5.1% of the words with actual roles as 'Object' were misclassifed as 'Agent' and 13.5% as 'No Role'. On the other hand, only 0.2% the words with actual role 'No Role' were wrongly predicted as 'Agent'.

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

While comparing the predictions made by the model variant in both the configurations, it was observed that the roles 'Agent' and 'Object' made most of error in predictions as compared to other roles. When using GF sentence form and localist word vectors, the roles 'Agent' and 'Object' were wrongly predicted as 'Recipient', whereas while using word2vec vectors the model, this misprediction do not exist. Comparing the false predictions made by model variant in both the configurations, it was also observed that in configuration-1 (while using word2vec word vectors), the incorrect prediction were comparatively less. In configuration-1, all the words with roles 'Recipient', 'Predicate' and 'No Role' were predicted almost without any errors whereas in configuration-2, the model variant misclassified these roles respectively by 16.2%, 36.4%, and 12.1%.

6.1.3 Experiment-3: Effect of Corpus structure

Word2Vec-ESN model: As illustrated in table 6.1, it was observed that when using the scrambled corpus as an input to the Word2Vec-ESN model, the model learned with low error rates of $ME = 6.68\%$ and $SE = 26.80\%$ in SCL mode, but while testing, the model generalized with cross-validation error rates of $ME = 70.15\%$ and $SE = 99.26\%$. Similarly, in SFL mode the model learned with low error rates of $ME = 9.38\%$ and $SE = 35.58\%$ but during cross-validation high error rates of $ME = 67.87\%$ and $SE = 99.26\%$ was observed. Notice that the model learned the scrambled corpus with low error rates while training but during the cross-validation resulted into high error rates.

Table 6.3: Training and testing classification scores for individual roles when using Word2Vec-ESN model variant in two different configurations.

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

Training and cross-validation classification scores (precision upto 2 decimals) for each output roles predicted by the model variant 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.

6.1.4 Experiment-4: Effect of Reservoir size

Word2Vec-ESN model: Figure 6.2 shows the effect of reservoir size on the cross-validation errors rates (i.e. meaning and sentence error). The meaning and sentence error continuously drops with increase in reservoir size from 90 to 1098 in both the learning modes. The meaning error in SFL mode slightly increases ($\approx 1\%$), whereas it remains unchanged in SCL mode as the reservoir size is further increased from size 1098. On the other hand, the sentence error in SFL mode slightly increases whereas in SCL mode it is negligibally dropped when the reservoir size is further increased from 1776 to 2882 and remains stable after that. Overall, it was observed that both the meaning and sentence cross-validation error rates reduces with increase in reservoir size but asymptotes when the reservoir size is above 1000 on corpus-462.

Word2Vec-ESN model variant: Figure 6.3 shows the change in classification scores of model variant with increase in reservoir size. It can be observed that when using the word2vec vector (Configuration-1) the classification scores sharpely 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 remains stable with neglible drop. Whereas, in Configuration-2 (i.e. when using GF and Localist representation) the classification scores also improves, with increase

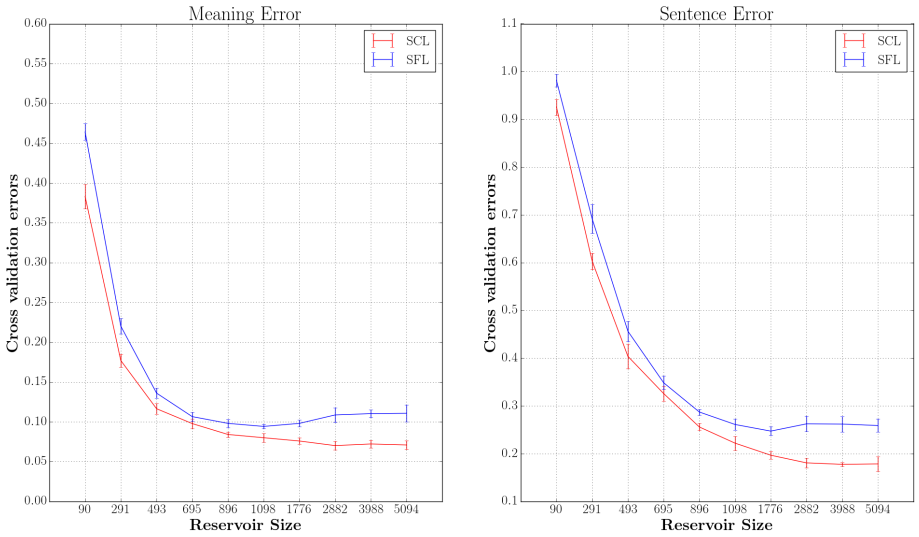

Figure 6.2: **Effect of reservoir size on cross validation errors on Model Variant-1:** Description goes here.

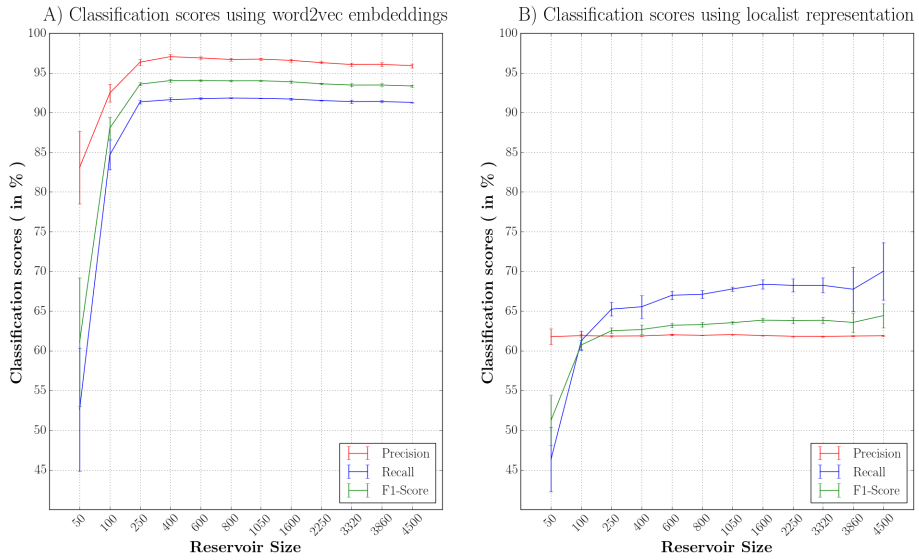

Figure 6.3: **Effect of reservoir size on classification scores of Model Varinat-2:** Description goes here.

in reservoir size from 50 to 250 neurons. As the reservoir size is further increased from 250 neurons, the recall is improved whereas the precision remains almost flat with neglibile improvement. As the F1-Score is harmonic mean of precision and recall, it was also slightly increased. Even the highest F1-Score of model variant observed at reservoir size 4500 in configuration-2, is much lower than that of obeserved at reservoir size 100 ($\approx 29\%$) in configuration-1.

6.1.5 Experiment-5: Effect of Corpus size

Figure 6.4 shows the cross-validation errors rates with respect to corpus size on Word2Vec-ESN model. It can be observed that with increase in corpus size from 6% to 50%, the meaning error sharply drops from some 12.18%(±0.19%) to 2.96%(±0.04%) in SCL mode and from 11.96%(±0.35%) to 4.17%(±0.11%) in SFL mode. Similarly, the sentence error also decreases from 54.84%(±0.52%) to 17.25%(±0.39%) in SCL and from 54.14%(±1.28%) to 22.41%(±0.57%) in 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%(±0.03%) meaning error and 15.98%(±0.30%) sentence error in SCL mode and with 3.95%(±0.11%) meaning error and 21.33%(±0.64%) sentence error in SFL mode.

However, it was also observed that, although the error rates gradually drops when the corpus size is further increased from 25% to 100% but the improvement rate of errors is very less. Thus further increase in corpus size wont have much effect on cross validation error.

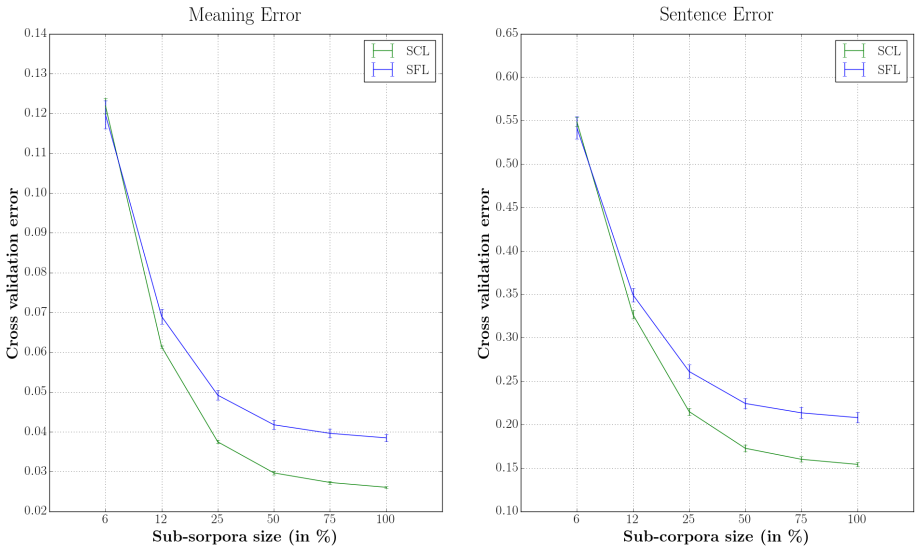

Figure 6.4: **Effect of corpus size on cross validation errors of Word2Vec-SN model in SCL and SFL mode.**

Comparing the effect of corpus size on Word2Vec-ESN model with that of $\theta RARes$ model (see fig. 6.5), it was observed that in SFL mode, with increase in

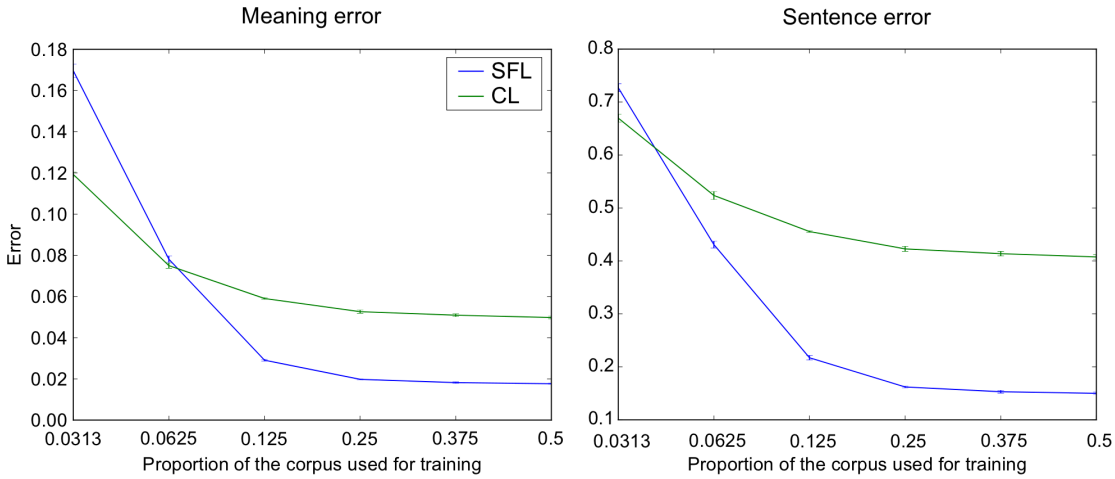

Figure 6.5: **Effect of corpus size on cross validation errors using localist word vector as reported in [ref]:** Description goes here.

corpus size from 6% to 25%, the meaning error in $\theta RARes$ model dropped from $\approx$ 17% to $\approx$ 4% and sentence error dropped from $\approx$ 70% to $\approx$ 20%, whereas in Word2Vec-ESN model meaning error dropped from $\approx$ 12% to $\approx$ 5% and sentence error dropped from $\approx$ 55% to $\approx$ 26%. Also, with increase in corpus size further from 25% the cross-validation errors asymptotes in both the models with negligible improvement in error rates. Overall, the improvement in error rates with $\theta RARes$ model in SFL mode, is higher as compared to Word2Vec-ESN model.

Comparing the performance of both models in the SCL mode, the drop in meaning error almost remained equivalent with increase in corpus. Whereas, with increase in corpus size from 6% to 100%, the sentence error in $\theta RARes$ model dropped from $\approx$ 65% to $\approx$ 40% and in Word2Vec-ESN model this drop was observed from $\approx$ 55% to $\approx$ 15%.

6.1.6 Experiment-6: Generalization on new corpus

In the Table 6.4, the sentence and best error values obtained with Word2Vec-ESN and $\theta RARes$ model are reported. The best error here represents the percentage of sentences whose meanings were predicted incorrectly in common within 10 model instances [16]. The Word2Vec-ESN model with a reservoir of size 500 neurons, generalized with a sentence and best error of 42.65% and 26.54% respectively. Whereas with a reservoir of 1000 neurons, the Word2Vec-ESN model generalized with 40.29% and 25.73% sentence and best errors respectively. In comparision to $\theta RARes$ model, the mean sentence error in Word2Vec-ESN model improved by 26.31% and 17.97% with resevoir of size 500 and 1000 respectively. The best errors also improved by 17.96% and 9.12% with reservoir size 500 and 1000 respectively.

Although, the Word2Vec-ESN model generalized better with both the reservoir sizes (500 and 1000) as compared to $\theta RARes$ model, but the improvement in cross-validation error in Word2Vec-ESN model is comparatively less than $\theta RARes$

Table 6.4: Generalization error in SCL mode on corpus-373.

Reservoir	Error	Word2Vec-ESN	$\theta RARes$
500 N	mean(std.)	42.65 (± 1.36)	68.96 (± 2.03)
	Best	26.54	44.50
1000 N	mean(std.)	40.29 (± 1.13)	58.26 (± 1.37)
	Best	25.73	34.85

Sentence and best errors (in %) obtained with Word2Vec-ESN model and $\theta RARes$ in SCL mode with reservoir of size 500 and 1000 neurons. The results reported are mean and standard deviation of errors obtained from 10 reservoir instances. The best error here means the percentage of sentence errors common within all 10 reservoir instances [16]. Simulation conditions for Wor2Vec-ESN model: Leave-one-out cross validation, SR=[] , IS=[] , LR=[].

model. The mean sentence error and best error improved by 2.36% and 0.81% respectively with increase in reservoir size in Word2Vec-ESN model. In $\theta RARes$ model, increasing reservoir size from 500 to 1000 leads to an improvement of 10.7% mean sentence error and 9.65% best error.

6.1.7 Experiment-7: Effect of Word2Vec word dimensions

Word2Vec-ESN model: Figure 6.6 and 6.7 represents the effect of word vector dimensions on cross-validation errors of Word2Vec-ESN model in SCL and SFL mode respectively. The corresponding errors are also reported in Table B.2. In the figures we can see that from lower to upper limit of word vector dimensions studied, all the error measures in both the learning modes increased approximately by 2%. Also, the cross-validation errors remained almost equivalent with negligible fluctuations in the range 20 to 200 dimensions, with the mimimum errors obtained with word vectors of 30 dimensions ($ME = 14.23(\pm 0.53), SE = 43.46(\pm 0.70)$). When the word vector dimension is further increased to 300, we can see that all the error measures increased nearly by 2%. Another interstin pattern which can be observed is that with all the word vector dimensions the model in SCL mode always performed better as compared to SFL mode. The similar relation between SCL and SFL model was also observed in previous experiments as well. Over we can say that the word vectors dimensions have negligible effect on the perofamance of Word2Vec-ESN model.

6.1.8 Neural output activity of the Word2Vec-ESN model

In the previous experiments, we observed that both the model variants generalized well and cross-validation error rates dropped with increase in corpus size. The 45

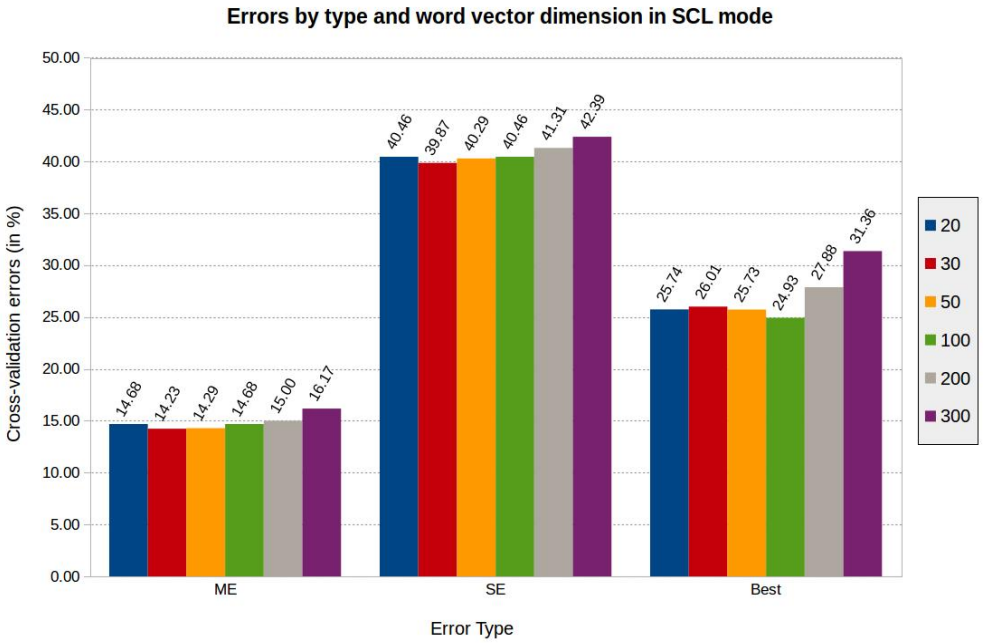

Figure 6.6: Effect of word vector dimensions on cross validation errors of Word2Vec-ESN model in SCL mode.

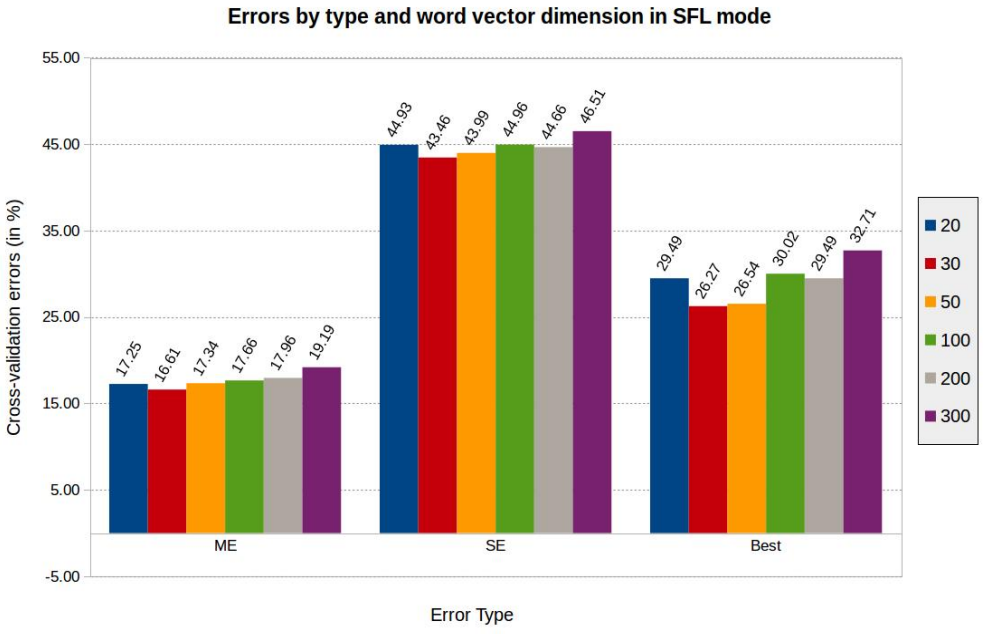

Figure 6.7: Effect of word vector dimensions on cross validation errors of Word2Vec-ESN model in SFL mode.

sentences of corpus-45 was added to corpus-462 to get the resultant corpus have 507 sentence (462 + 45). The readout activations of the Word2Vec-ESN model were then analysed for the input sentences to have the insight of how the system is learning the meaning. It was observed that the model is re-analyzing the thematic roles of input sentences across the time. The same behaviour was also observed by Hinaut et. al [16, 15] with $\theta RARes$ model.

Figure 6.8 shows the read out activations of second noun in following four sentences across time. Note that the sentences 1 and 2 with 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).

Each coloured lines in the graph represents the possible thematic roles of Noun-2 (N2), which can have one of the three possible roles i.e. agent (A), object (O) or recipient(R) with resepct to either Verb-1 (V1) or Verb-2 (V2).N2 is marked in red and verbs are marked in green. In the figure the role 'N2-A-V1' can be interpreted as noun-2 is the agent of verb-1.

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

With the arrival of the V1 ('took'), in sentence 2, the activation for N2 as an agent of V2 goes above threshold indicating the presence of V2 ('hit') in the sentence even before the model has encountered the second verb. Whereas in sentence 1 the activation of role N2 as agent of V1 maintained and no other roles managed to cross the threshold with the arrival of V1 ('gave'). In sentence 3 and 4, with the arrival of V1 ('caught' and 'pushed') one can notice the re-analysis made by the model, with activation of N2 as agent of V1 going below threshold and taking the new role as object of V1 ('gave') in sentence 3. Also, in sentence 3, the activation of role N2 as object of V2 (N2-O-V2) goes above threshold whereas in sentence 4 the previously predicted role N2-R-V1 goes below threshold with the arrival of V1 ('pushed').

Throughout the rest of sentence 1, the prediction N2-O-V1 is maintained with some minor ups and downs. In sentence 2, the predictions of N2 alternates between O-V1 and A-V2. While the arrival of "the" and "ball" seems to favour O-V1, arrival of "that" changes the prediction again to A-V2. In both cases, the activation of

both the role 0-V1 and O-A2 remains above threshold. In sentence 3 the model sees both O-V1 and R-V1 as the prefered predictions, both being almost on the same level and only alternating slightly . In sentence 4, the prediction of role N2 as agent of V1 do not change again throughout the sentence ever since arrival of V1 ('pushed').

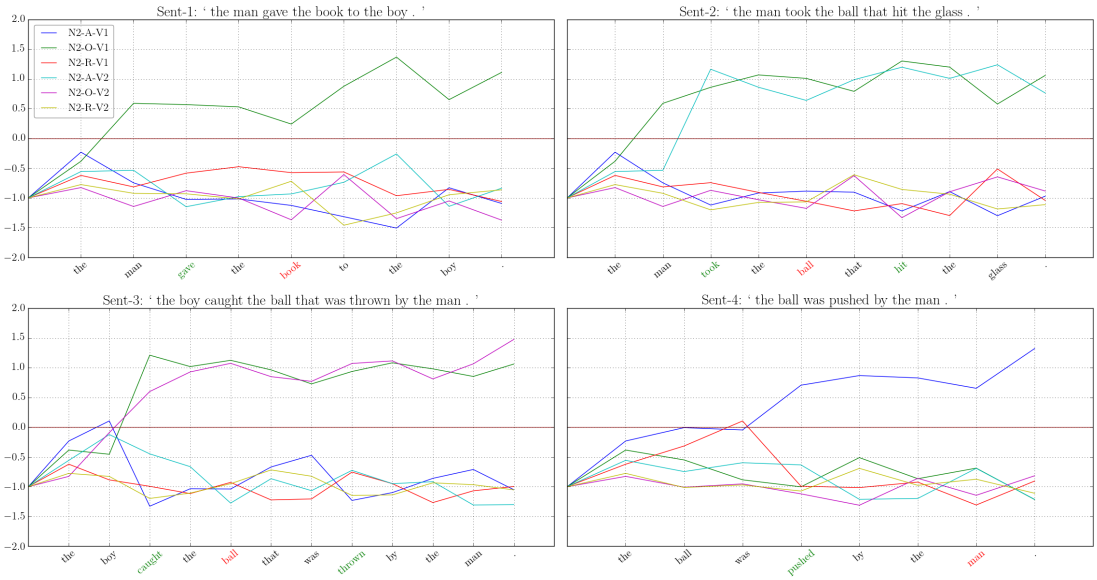

Figure 6.8: **Short discourse processing by Word2Vec-ESCN Language model:** Each coloured line shows the thematic role of Noun-1 (tick marked in red) with respect to Verb-1 and Verb-2 (ticks shown in green). The coded meaning for the second segement of sentence is resolved using the information from first segment of sentence. Thus, also resolving the anaphoric reference to 'he' and 'it'. Model predictions: A) N1 is agent of verb-1 and verb-2. B) N1 is agent of object of verb-1 and verb-2. C) N1 is agent of verb-1 and object of verb-2. D) N1 is object of verb-1 and agent of verb-2.

Discourse processing: Figure 6.9 shows the read out activations of Noun (N1) in the following sentences with two-sentence discourse segment across time.

(A) *John*(N1) *threw*(V1) the ball. Then he *caught*(V2) it.
(B) The *ball*(N1) was *thrown*(V1) by John. Then he *caught*(V2) it.
(C) *John*(N1) chased the dog. Then it *bit*(V2) him.
(D) The *dog*(N1) was *chased*(V1) by John. Then it *bit*(V2) him.

Note that the first segment of sentences (A) and (C), (B) and (D) have the same grammatical form, whereas the second part of the senteces (A) and (B), (C) and (D) have the same grammatical structure.

One can see in the figure 6.9 that the model was able to resolve the roles for second verb (V2) present in second segment of all the sentences, using the

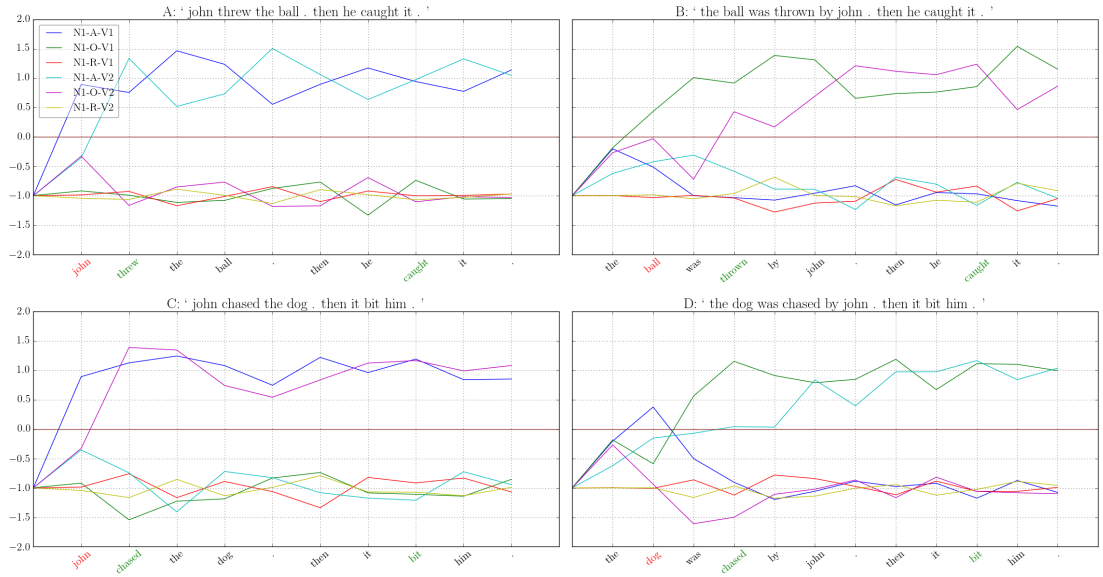

Figure 6.9: **Effect of corpus size on cross validation errors using localist word vector as reported in :** Description goes here.

information from first part of the sentences, although there were no explicit nouns in the second segment of sentence. Thus also the anaphoric reference to 'he' and 'it' is also resolved.

Comparing sentences (A) and (B), both have the same second segment i.e. "then he caught it", but model predicted different coded meaning for both this segment of sentence, depending on the first segment of sentence. In (A) the model makes an eary prediction that N1 is the agent of the second verb ('caught'), whereas in sentence (B), N1 is predicted as the object of the second verb. Similarly, the senteces (C) and (D), the second segment is same in both the sentences i.e. " the it bit him", but the model predicted the N1 is object of V2 ('bit') in sentence (C) and N1 is agent of the same second verb.

Also, notice that the roles for N1 with respect to V2, is predicted earlier in sentences (A) and (D) as comared to that of sentence (B) and (D), where in the latter the presence of V2 and the role of N1 with respect to V2 is confirmed only after the arrival of V1. With the arrival of 'was' and V1 ('chased') in sentence (D), the model changes its previous prediction that N1 is agent of V1 and assigns a new role that N1 is object of V2 ('bit').

Chapter 7

Conclusion And Future Work

7.1 Conclusion

To be written...

7.2 Future Work

For this work we currenly used the combination of word2vec model and ESN. The Echo state network has an advantage of modelling sequential data, thus the sequential and temporal aspect of a sentence is taken into account in this study for thematic role assignment. But the dependencies between thematic roles of sentences were not taken into account for learning. To model the conditional probality distrubtion of the thematic roles, Conditional Random Fields (CRF);a log-linear model; can be used [40]. CRFs has been the one of the most successful apporach used earlier as well for classification and sequential data labelling tasks [47, 10]. Thus the Word2Vec-ESN model, proposed in this study, can be used with an additional CRF unit at the end to model the temporal dependencies between the input sentences conditional on the corresponding themactic roles. Doing so allows the resulting model to capture concealed temporal dynamics present in the sentences[10].

Also several low dimensional word vector can be generated using the word2vec model. It was observed that with increase in word vector dimensions the accuracy on Semantic-Syntactic word relationship test set [42] also increases till some point. However adding more dimensions results into reduced improvement [29]. Although in the current study we used word a vector 50 dimension but the effect of other dimension were not explored. It would also be intersting to explore the effect of higher word vector dimension on the performance of Word2Vec-ESN model. Word2Vec word vectors obtained by training the model on one language corpus (say English) can also be translated to get the most similar word in any target language. This is achieved by linearly projecting the word vector of source languge on target language [30]. Thus the Word2Vec-ESN language model can also be investigated for multiple language acquisition [17].

Appendix A

Nomenclature

Appendix B

Complete Simulation Results

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

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

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

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

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

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

Bibliography

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

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

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

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

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

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

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

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

[32] 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.

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

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

[35] 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.

[36] 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.

[37] 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`.

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

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

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

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

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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