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?

Zusammenfassung

Hier die deutsche Zusammenfassung einfügen (notwendig).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Thematic Role Assignment (TRA) is a supervised learning problem which aims to identify events and its participants 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 [42]. Understanding the semantics of the text plays an important intermediate step in a wide range of real-world applications such as machine translation [24], information extraction [4], sentiment analysis [41], document categorization [30], human robot interaction [15, 4] etc.

1.1 Previous Work

Many successful traditional system consider SRL as a multiclass classification problem use linear classifier such as Support Vector Machines (SVM) to tackle the problem [23, 32, 31]. These system 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 syntatic parser certainly leads to degradation of predictions [31]. Also designing of feature templates need a lot of heuristics and time. The pre-defined features are often required to be iteratively modified depending on how the system perfoms. The feature templates are often required to be re-desinged when the task is to be performed on different languages, corpus or when the data distribution is changed [42].

In order to avoid engineering manual feauture templates, SRL task was also attempted with neural network models. Collobert et al. [11] first attempted to build an end-to-end system without parsing by using word embeddings and Convolutional Neural Network (CNN). The model was less successful as CNN cannot

employs long term dependencies within a sentence since it can only take into account the words in limitied context [42]. However to increase the model perfomance they also resorted to use syntatic features by using parse trees of charnink parser [9].

Recurrent Neural Networks (RNN) has been also been used for wide range NLP task and also recently with Echo State Network (ESN): a varinat of RNN. The tasks used were diverse from perdicting next word given the previous words to learning grammatical structures [39]. RNN makes use of sequential information and acts as a memory unit and captures the information processed in the past [12]. 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 in ESN the recurrent layer (reservoir) is randomly initialized and only connections from recurrent layer to read-out layers are learned [21, 25]. These advantages of ESN over RNN makes it good choice to be used for TRA task.

Xavier et al. [14] proposed a generic neural network architecture 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 brain and provided a robust and scalable implementation on robotics architecture [14, 15]. They called this model as $\theta RARes$. 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 system they also treated words as discrete atomic symbols and used localist vector respresentaion of words as an input. Treating each word as a dicrete 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, then the semantic relationship (i.e. both are colors) between these two words is lost [1]. 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 on the other hand the 'SW' token in iteself does not carry any semantics and thus does not allow 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 more in

detail about the limitation of this model later in Chapter 3.

Motivated by the limitations of localist input representation of words and transformation of raw sentences into its abstract form by replacing semantic words with 'SW' token described above, we hypothesize that the use of distributed word representation which can capture the syntactic and semantic relationship of words could possibly improve the perfomance of the model on TRA task. One such model for learning distributed word vectors was proposed by Mikolov et al. [26] widely known as Word2Vec model. Word2Vec model learn high quality, low-dimensional vector represenation of words from a large corpus in an unsupervised way[ref-two]. The resulting word vector of this model encodes semantics of words. As the model learns the embeddings by taking into account the context words, the obtained vector embeddings also encodes several language regularities and patterns[ref] and can be observed by performing linear operation on the word vector. For example, *vector('king') - vector('man') + vector('woman')* $\approx$ *vector('queen')*. Unlike other neural network models for obtaining word embeddings, training Word2Vec is computationally cheap and efficient[ref?]. Training of word2vec model and properties of resulting distributed word embedding will be discussed in detail in Chapter 2.

1.3 Proposed Models

In this work, we thus propose a end-to-end system called *Word2Vec-ESN* model, for TRA task. The Word2Vec-ESN model is a combination of word2vec model and ESN. The word2vec model is trained on a general purpose unlabelled dataset (e.g. wikipedia) prior to use of model for TRA task. The word2vec unit being the first unit, receives the raw sentences and generates the distributed word embeddings of the constituent words. 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 basically a modified verision of $\theta RARes$ model [14], where unlike the latter raw sentences are not transformed to grammatical construcion and word2vec word vectors are used over localist word representation as an input to ESN.

Apart from Word2Vec-ESN model, we also propose a variant of this 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 feautre to this model variant is the current word and the verb with respect to which it is processed. The output units encodes the possible role (e.g. predicate, agent, object, recipient and No Role) of the input words unlike the original model where output units encodes the thematic roles of all sematic words in the input sentence. For the evaluation of this model variant we used metrics (classification scores) proposed for CoNLL-2005 SRL task[ref]. Both Word2Vec-ESN model and its variant is discussed in mored detail in chapter 4.

[TODO:Describe the overview of results here.]

1.4 Scope of work

There are several other ways of obtaining word embeddings [glove and other] but a systematic comparision of them on TRA task is beyond the scope of this work. Using word2vec model, distributed word vectors of different dimesions can be obtained and used for TRA task. Evaluating and comparing the effect of dimesnsion of these word embedding is also not the focus of this study. To this date, there is no research conducted with this combination of word2vec model and ESN. This also makes this study novel.

1.5 Outline

In the next chapter we give a description of word2vec model and echo state network model. We also descibe in detail the training of word2vec model and properties of word2vec word vectors. Alse this chapter describes training and control parameters of ESN. The chapter 3 describes the $\theta RARes$ model, its limitations and the motivation and hypothesis for the current work. Subsequently in chapter 4 we propose the Word2Vec-ESN model and its variant. This chapter also describes the data, implementation and evaluation metrics used in our experiments. Chapter 5 contains the experiments and results performed for TRA task with proposed model along with the results. This chapter we also compares the results of Word2Vec-ESN model with the results obtained from $\theta RARes$ model and analysis them. Finally, in the chapter 6 we describe the conclusion of this study and the possible future work.

Chapter 2

Basics of Word2Vec and Echo State Network

2.1 Word2Vec

Word2vec is a neural probalistic language model based on Distributional Hypothesis which states that the words that appears in the same context share the semantic meaning [1]. The proposed by Mikolov et al. [26], which takes in input a large text data to generate the distributed word embedding of the words present in 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 closed to each other in the vector space. Earlier the words have been treated as discrete atomic symbols in all traditional NLP system, where each word were represented in a localist fashion. Localist representation of words does not contain any semantic or syntactic information of the word it is encoding and thus depriving the NLP systems to utilize this information while processing [3]. Word2vec neural word embeddings overcomes this issue and capture the semantics and syntactic information of the word in a computationally-efficient manner [27]. For learning word embedding two neural architecture were proposed, the Continuous Bag Of Word (CBOW) and Skip-Gram (SG) [27, 26]. 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 skip-gram model in our work because it was proven that it produce better word-embeddings as compared to CBOW [26].

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 ball to). Figure 2.1 shows the CBOW network architecture with a simplified case of one context word. The model is trained on the dataset having a vocabulary size V in an unsupervised

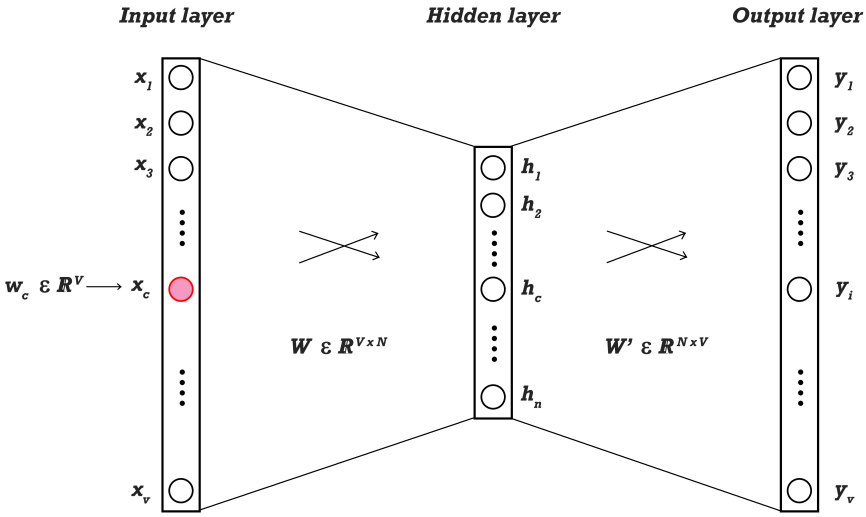

Figure 2.1: **The CBOW model:** In the CBOW model, the objective is to predict the target word from the words in the context (or neighbouring words). The context words are input to the model (in this case only one) using localist representation where only one vector element corresponding to input word 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 [ref].

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 [34]. 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 $W^{'}$. The computed scores are then converted to posterior probabilities by output

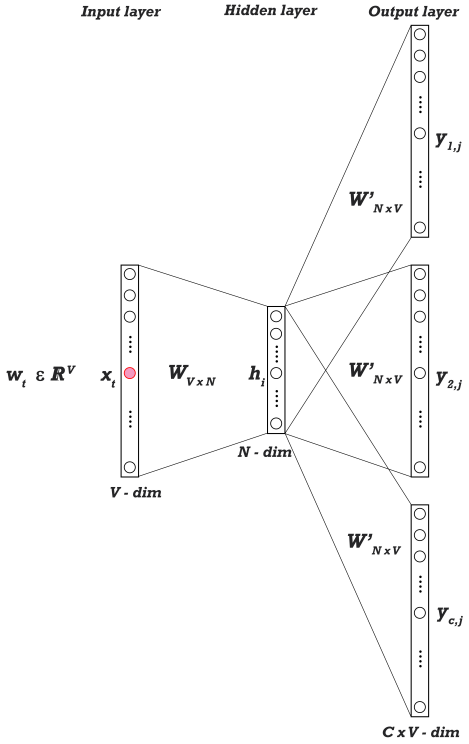

Figure 2.2: **The skip-gram model:** In the skip model, the objective is reverse of CBOW. It predict the context words from the word. The target word is input to the model using localist representation where only one vector element corresponding to input word is active (x_t, shown in red). The model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

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 [34].

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 [26]. 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 need 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[26].

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 [26, 1]. 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 $-c \leq j \leq c$.

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 [28, 26]. 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 encodes semantic and syntactic properties of the words as a constant vector offset between a pair of words sharing a specific relationship[26]. For example, the word embeddings "$King - Queen \approx man - woman$", "$apples - apple \approx cars - car$", "$walking - walked \approx swimming - swam$".

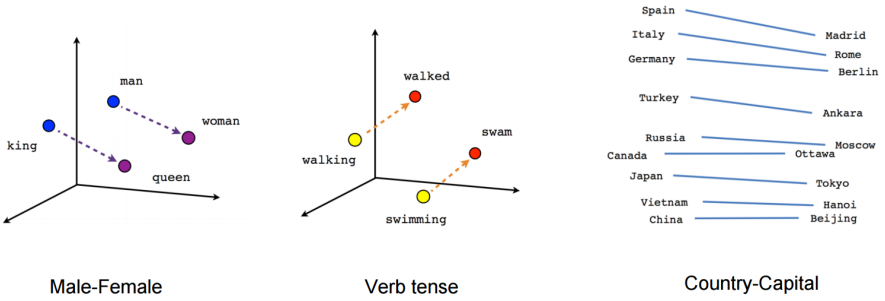

Figure 2.3: **Word2vec semantic regularities.**

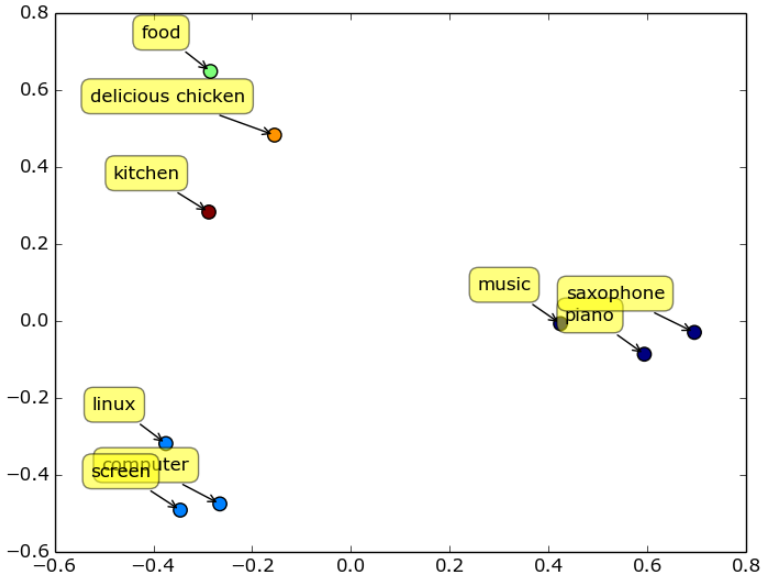

Figure 2.4: **Word Clustering with word2vec word embedding:** The figure shows the word clustering property obtained by projecting the word vectors on two dimesional space using PCA. The words vector taken from pretrained word2vec Google News corpus[1]

[2]

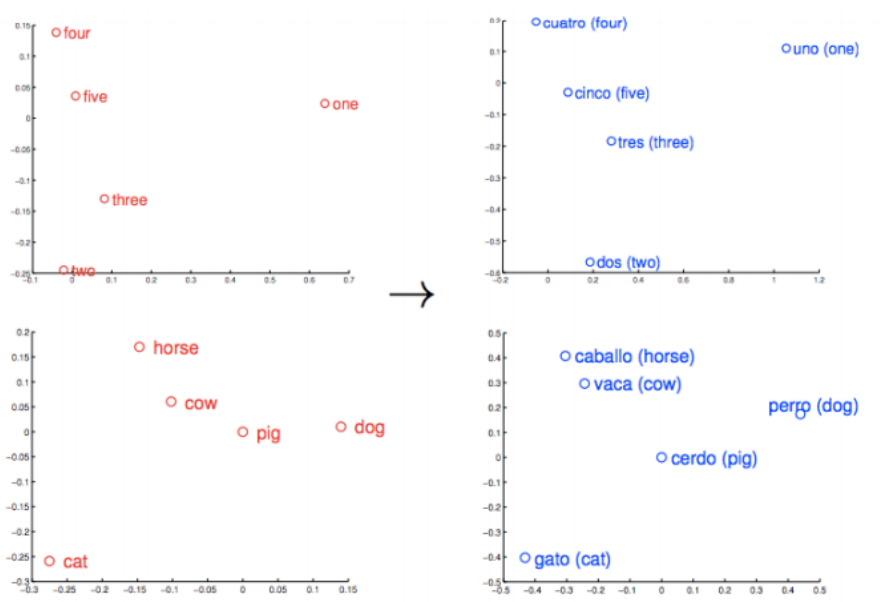

Figure 2.5: **Word2Vec language translation property.**

The another interesting property is that the 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 of respective language. Thus it is also possible to learn linear mapping between different embedding space by vector rotation and scaling [28]. Several other regularities can also be captured by performing basic linear operation on word-embeddings [26].

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 [20] 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 [19, 20, 18].

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

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 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 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 [20, 25]. 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 word 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 [25, 19].

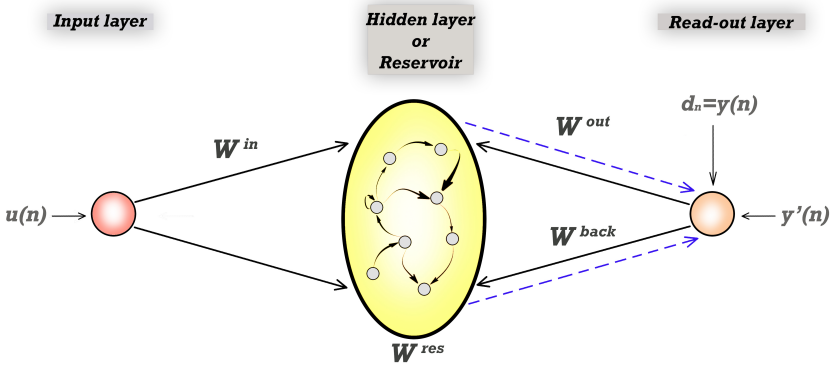

Figure 2.6: **Architecture of classical ESN:** The reservoir is the recurrent neural network with N_x units and initialized with random connection. The reservoir is provided input $u(n)$ to the input layer and teacher layer $y(n)$ are pushed output neurons respectively during training. The input to reservoir weights (W^{in}), output neurons to reservoir (W^{back}, optional and depends on task) and reservoir to reservoir weight (W^{res}) from are also randomly initialized and stays static during learning. The output weight from reservoir to output unit are the only weights learned by the network during training. Adapted from [25]

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 [25]. The input weight matrix is however a bit more 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 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 [19].

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

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 $\rho(W^{res}) > 1$ [20, 25, 19]. 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 to behave in highly non-linear manner (because of tanh activation function) whereas a very small input scaling is used wherever linearity is required in a task [25].

2.2.2 Training ESN

ESNs are mostly applied supervised machine-learning tasks where temporal or sequentail 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 linear combination with the desired output signal [25, 19]. 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 $u(n)$ [19]. 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. Smaller value of also induces slow reservoir dynamics thus ensuring the long short-term memory in ESN [25, 22]. 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.

Training procedure of ESN, have only few global parameters which are to be optimized: 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.

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

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. [14] proposed a neuro-inspired model, $\theta RARes$, to process the sentence across time without having to know the sematics of the words. The model used reservoir computing approach namely echo state network and implemented on robotics architecture [15] for the thematic role assingment task. This was the first time when echo state network was used for thematic role assingment task. The experiments done for TRA showed the results toward modelling of language acquisition in brain [17, 14]. 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. This principle was then utilized for thematic role assingment task.

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 the mapping of surface form (word order) of sentence to its meaning [13].Figure 3.1 represents the characterization of thematic role assignment in grammatical construction form. The model does not takes in input the raw sentences but instead use the abstract form of the sentences. The abstraction marks each word of a sentence into two kind 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, verb inflexions 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).

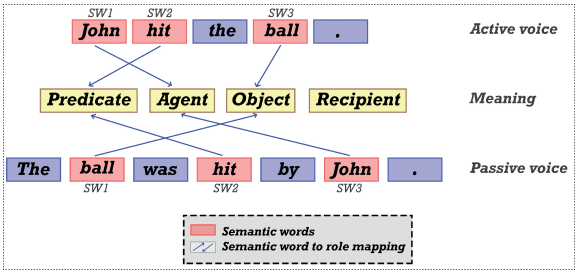

Figure 3.1: **Word2vec semantic regularities.**

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

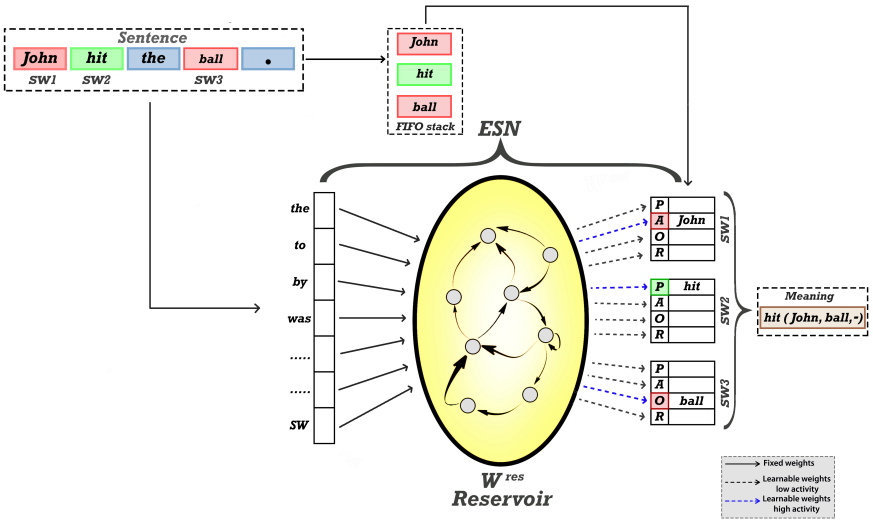

Figure 3.2: **Word2vec semantic regularities.**

Thus for training the transformed sentences were presented to the model sequentially, 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 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 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 for such an input representation mainly occurs with the ambiguous sentences. A sentence is said to be ambiguous if it has the same grammatical construction but different coded meaning and actual meaning. 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		

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 localist word vectors as an input to the ESN have produced the promising results for the thematic role assignment task[14, 17]. But the behaviour 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 [26, 38]. 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 neighborhood within words embedding space (see Fig. 3). This could possibly allow the model to learn this 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 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 embedding is 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 have 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 inpired from the $\theta RARes$ model proposed by Hinaut et al. [14] for TRA task. Word2Vec-ESN model is basically the combination of Word2Vec model and Echo State Network (ESN). Figure 4.1) shows the architecutre 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 input to ESN, which further processes these embeddings to learn and predict the thematic roles of all semantic words in the input sentences.

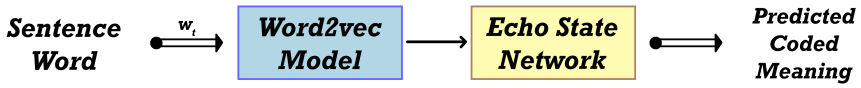

Figure 4.1: **Architecture of Word2Vec-ESN model:** The model takes the words of the input sentence as an input across time. Word2Vec model generates the distributed vector representation of the input word. The generated word vector is then used by ESN for further processing and learns to predict the 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 [26] approach on a general purpose dataset (e.g. Wikipedia) and the domain specific dataset. During the training the word2vec model learns the the low dimensional distributed embeddings for each word in the corpus vocabulary (see section 4.4).

The reservoir in ESN, composed of leaky integrator neurons and sigmoids activation fuction, was sparsely and randomly initilizied. A fixed fan-out of 10 and 2 is chosen for hidden-to-hidden and input-to-hidden connections respectively. In other words, each reservoir neuron is connected to 10 other reservoir neurons and each input neuron is connected to only 2 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 [18, 25].

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 [14]. Thus it enable us to compare the performance of both the model.

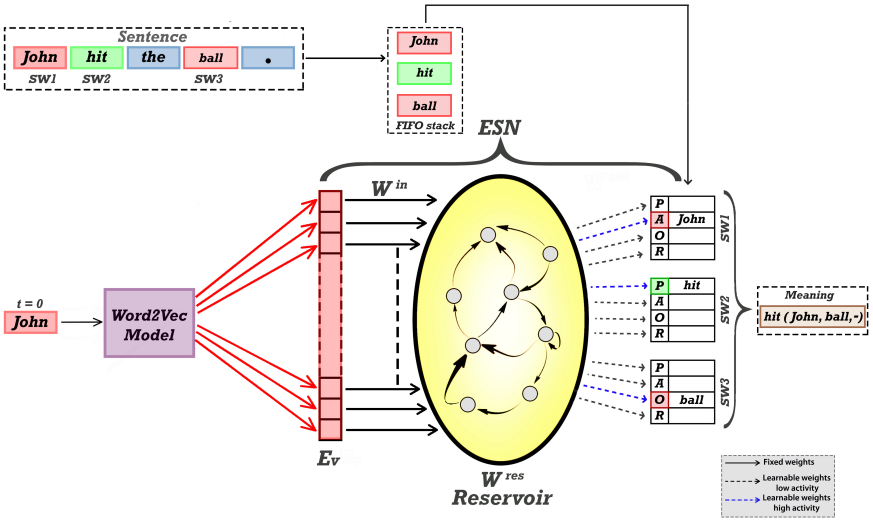

Figure 4.2: **Word2Vec-ESN language model:** The figure shows the processing of a sentence by the model variant at time step 1. Nouns and verbs (specified in red and green respectively) are stored in a memory stack for interpretating coded meaning. The word *'John'* is input to word2vec model which generate 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 codes 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 [14]

Figure 4.2), shows the neural comprehension fo Word2Vec-ESN model for thematic role assignment.

During the training, sentences are presented to the model one at 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 reaout 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 (e.g. nouns and verbs) 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), Receipient (R). For example, if the sentences in the corpus have maximum of N_{sw} sematic 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 output neurons have an activation of 1 if corresponding thematic role is present for the sentence, -1 otherwise. The Word2Vec model receives the words from the input sentence across time and generate the 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 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.

This model variant can be operated in two learning modes, so that it learns to extract the coded meaning of each noun with respect to a verb.

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 first word of the sentence. Thus, the regression is applied with teacher roles from the onset of first word in the sentence [14].

2. **Sentence Final Learning (SFL)**: In this mode the learning takes places only at the end of the sentence [14]. Hence, the teacher labels are only provided to the network at the end of sentence i.e. from last word of the sentence to the final period.

Decoding Output: As described earlier, coded meaning of 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 maximum of all activation between 4 possible roles (i.e. Predicate, Agent, Object, Recipient) is taken as the coded meaning of a semantic word [14]. A semantic word is said to have incorrect coded meaning if the winning readout neurons is not the correct role of the semantic word. If there is no activation above threshold for a

semantic word, then this semantic word is considered to have no coded meaning [14]. 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.[14] 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 [14]. 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 [14] 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 4.3 illustrates the functional orgainisaion 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 plays an important role in this model variant: argument and predicate, with 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 [42].

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

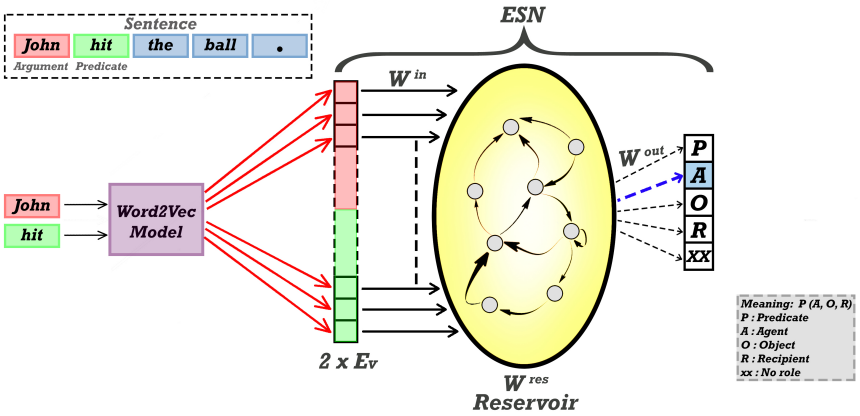

Figure 4.3: **Word2Vec-ESN Model Variant:** The figure shows the process of a sentence in model variant at time step 1. At any instant of time an argument (current word, marked in orange) and predicate (verb,marked in green) is input to the model. Word2Vec model generates the word vectors of E_v dimensions which are then cocatenated 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 (i.e. A (Agent)). The read-out weights (shown in dashed line) are learned during training. During testing the readout unit codes the role of input words, which are then accumulated and decoded to meaning *hit(John,ball,–)* at the end of sentence. Inspired from [14]

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 have 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 have the corresponding role, -1 otherwise. The Word2vec model initally receives the 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 4.3 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. 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 4.3, 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 contigiency table [2] 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 of chosing macro-average is that it gives equal weights to all the roles addressing the role imbalance problem [29]. Higher the classification scores the better. The same evaluation metrics was used for CoNLL-04 and CoNLL-05 semantic role labelling shared [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 corresponds to the actual roles and the columns corresponds the predictions made by the model. The diagonal elements of this matrix represents the number of words for which the predicted role is equal to the true role, whereas all the non-diagonal elements represents the number of word which were labelled incorrectly. As the values of diagonal elements of confusion matrix indicates number of correct predictions so higher the values of diagonal elements the better.

Using the confusion matrix accuracy can be calculated as ratio of number of correctly labelled words to total number of words (equation 4.2.1). This meaures specifies how often the classifier is correct [35].

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

However, accuracy measure can be distorting (becuse of accuracy paradox) when the dataset have 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 [37, 40]. In our dataset we have imbalanced roles as most of words have labels "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 [35].

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

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 labelling the correct roles. It is also called *'Sensitivity'* or *'True Positive Rate'*. [35].

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

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 [35] and is calculated as:

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

4.3 Dataset and pre-processing

4.3.1 Corpus For TRA Task

In this study we used the corpus with 45, 462 and 90582 sentences. The sentence in corpus-462 and corpus-90582 was generated by Hinaut et al. using a context-free-grammer for English language and used for TRA task [14]. Each sentence in these corpora have verbs which 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 [14]. 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'.

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. 4.4). In total there were only 2043 distinct coded meanings [14]. This corpus also have an additional property that along with complete coded meanings for sentences it also have incomplete meanings. For e.g. the sentence The Ball was given to the Man have no *'Agent'*, and thus the meaning of the sentence is give(-,ball,man). The corpus also contains 5268 pair and 184 triplets of ambiguous sentences i.e., 10536 and 553 sentences respectively. Thus in total there were 12.24% (i.e., $5268 \times 2 + 184 \times 3 = 11088$) of ambiguous sentences which have the similar grammatical structure but different coded meaning [14]. 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 is 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 appears in the raw sentence. The third part (between ';' and '#') is the raw sentence with verb inflexions (i.e. '-s') and the fourth part is the abstract representatino of sentence with semantic words removed and replace with 'X' [14].

We preprocessed these constructions, to obtain the raw sentences without verb inflexions. Firstly, all the words are lower cased and then the verbs with inflexion is replaced by conjugated verb. The verb conjugation to be used depends on the inflexions used for the verb. For example the sentece 'The giraffe walk -s' has been changed to 'The giraffe walks'. This preprocessing was done because we have distributed word representation which captures this syntactic relations e.g $walks - walk \approx talks - talk$. We also added additional token '$< start >$' at the

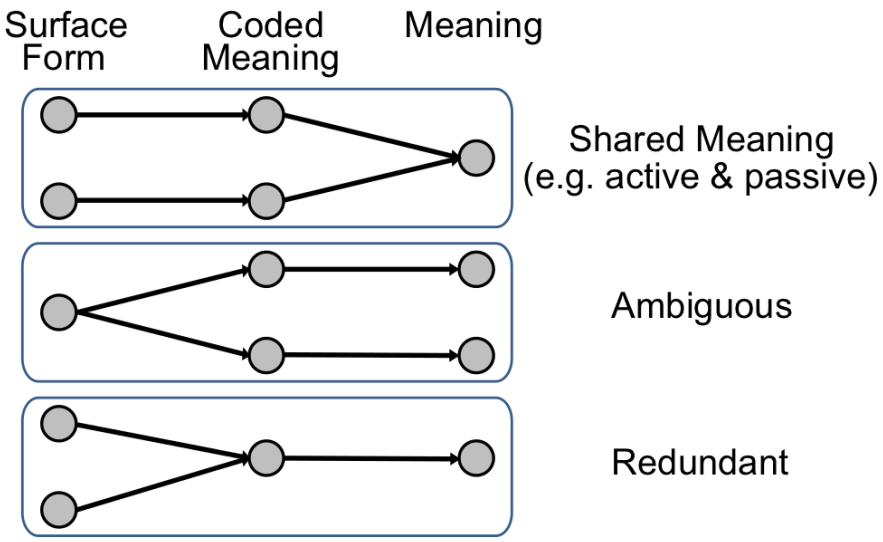

Figure 4.4: **Relation between surface form and meanings:**

beginning of sentence and '< *end* >' token at end, to mark the beginning and end of a sentence.

4.3.2 Corpus For Training Word2Vec Model

To train the word2vec model, we used wikipedia corpus (≈ 14 GB) to obtain the low dimesnional distributed of embedding of words. The corpus contains 2,65,8629 unique words. We chose to use Wikipedia data because we needed a gereral purpose dataset to have 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 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.

4.4 Obtaining Word Embeddings

To get the vector representation of words we first trained word2vec model on Wikipedia dataset. For training we used word2vec model with skip-gram negative sampling (see Chapter 2 for more details) approach to obtain the word embeddings as it is claimed to produce better representation as compared to CBOW approach and is easy to train [**?**, 26]. We used the hidden layer with 50 units (desired dimensions of word embedding), and a context window of ±5. The negative sampling size is chosen to be 5 i.e. 5 noise words are chosen randomly from the vocabulary which does not appear in the context of the current input word. We

ignored all the words which appears less than 5 times in the corpus. To update the network weights stochastic gradient is used. The initial learning rate was set to be $\alpha = 0.025$, which drops to $min_alpha = 0.0001$ linearly as training progresses.

The word embeddings obtained from training on Wikipedia dataset are good enough to capture the semantic relationship between words for e.g. $vec(Paris) - vec(France) + vec(Germany) \approx vec(Berlin)$. While training the model on Wikipedia data, a vocabulary of words is created and once the vocabulary is created it is not possible add new words to this vocabulary. However there is a possibility that when a domain specific corpus is used to further train the word2vec model some 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 [1] nor Gensim python API [33] implementation of Word2Vec supports vocabulary update once created. So, we implemented the online training[2] of word2vec by modifying and extending Gensim API. The new words not present in existing vocabulary is added and initialized with some random weights, which can then be trained in usual manner to have vector embeddings. Although now the vocabulary can be updated in online manner but the vector embedding of newly added word have poor quality if its count in new corpus is less. This can be improved by repetition of new dataset several times before training the model [3].

So now when we have an online version of training word2vec model, we extend word2vec model by resuming the training on the domain specific corpus (corpus-462 and corpus-90582). While updating the model on new dataset we do not disregard any words irrespective of the count, so that we have vector embeddings of all the words in our corpus. Once trained, the vector embeddings are normalized using L-2 norm before using them.

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

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

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

Chapter 5

Experiments and Results

In chapter we describe the experiments performed with the Word2Vec-ESN model and the variant proposed in the previous chapter. The results obtained by the Word2Vec-ESN model, using word embeddings, are compared with $\theta RARes$ model which uses sentence in grammatical form by replacing semantic word with 'SW' token and localist representation of words.

5.1 Input and Ouput Coding

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 input for ESN. Thus input layer have 50 neurons. For the experiment with corpus-462 we used a reservoir of 1000 leaky integrator neurons with *tanh* activation fuction. For the output coding the topologically modified but equivalent representation is used [14]. Thus, the readout layer contains 24 ($4 \times 3 \times 2$) neurons as the corpus contains sentences having maximum of 4 nouns each having 3 possible roles (Agent, Object and Recipient) with respect to a maximum of 2 verbs. Output neuron have an activation 1 if the role is present in the sentence, -1 otherwise. Whereas when using corpus-90582 for training the number of neurons in reservoir were raised to 5000 and also the readout neurons are increased to 30 ($5 \times 3 \times 2$) as there were maximum of 5 nouns in the sentences of this corpus.

Word2Vec-ESN model variant: In the 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 have 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 processed. The reservoir

of size 1000, 3000 is used for corpus-462 and corpus-90582 respectively. 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). Readout neuron of ESN have an activation 1 if the input word-verb (argument-predicate) pair have the corresponding role, -1 otherwise.

5.2 Experiments

5.2.1 Experiment-1: Learning thematic roles

In order to determine the model capability for predicting thematic roles of the sentences using word2vec embeddings for words, we first did the experiment using 26 sentences(sentence 15 to 40, in corpus-45) from corpus-45. The chosen sentences have distinct surface form (e.g. active, passive, dative-passive) and grammatical structure. This also include the sentences with single verb or double verb relative surface form [ref?]. Both the model variants(see section-?) learned the sentences without any error when trained and tested on all the sentences. To test the performance and generalization capabilities of model on untrained sentences, we performed a leave-one-out cross validation, where a model is trained on 25 sentences out of 26 and tested on remaining 1 sentence.

Model Variant-1: The model variant-1 with a reservoir of size 1000 units the model yielded [?] meaning error and [?] sentence error in sentence continuous learning mode and [?] meaning error and [?] sentence error in Sentence final learning mode. The results were averaged over 10 reservoir instances. For Continuous learning model the spectral radius(SR), input scaling (IS) and leak rate(α) were identified as $SR = [?], IS = [?], \alpha = [?].$‘ Whereas for Sentence final learning mode the $SR =?, IS =?, \alpha =?$. The ESN parameters for which the optimized results are identified are found by exploration of parameter space. As one may note that difference in training and test error for both meaning and sentence error is large. This indicates the model is overfitting on the dataset. However, it is not surprisring because the dataset contains limited examples, which constrained model to generalize well. As this experiments remains a toy demonstration we will explore the generalization capabilty of the model in section 5.2.2 .

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.

5.2.2 Experiment-2: Generalization Capabilities

In the previous experiment we demonstrated the performance of the model with the limited set of sentences where the results suggested that the model is probably

overfitting and not generalizing on the unseen sentences. So in order to test the generalization capability of the model, we examined the model's behaviour with an extended corpus of 462 sentences (see corpus-462 in 4.3) using 10-fold cross validation. 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 is trained and tested on all the subsets atleast once.

Word2Vec-ESN: We initially trained and tested the model with reservoir of 1000 neurons on all the 462 sentences. The model learned the full corpus-462 with 0.54% meaning error and 1.51% sentence error in SCL mode and 0.14% meaning error and 0.43% sentence error in SFL mode. Using the 10-fold cross validation, the model generalized to 7.82%($\pm$1.59%) meaning error and 20.65%($\pm$2.79%) sentence error in SCL mode with spectral radius (SR), input scaling (IS) and leak rate(LR) of 2.4, 2.5 and 0.07 respectively. Whereas in the SFL mode with $SR = 2.2$, $IS = 2.3$ and $LR = 0.13$, the optimal meaning and sentence error were observed as 8.68%($\pm$1.26%) and 23.69%($\pm$1.17%) respectively. The optimal parameters for both the learning models (SCL and SFL) were identified using grid search over the parameter space.

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 5.1, during testing, we observed an improvement of 11.48% sentence error in SCL mode with Word2Vec-ESN model whereas meaning error remained almost equivalent in both the models. In SFL mode, using Word2Vec-ESN model, both meaning and sentence errors dropped nearly by 1%. One can also notice that with Word2Vec-ESN model, the performance gain in more in SCL mode as compared to SFL mode. The reason is that the word-embeddings in word2vec model are learned from the context words and thus word vector encapsulates the information about neighbouring words.

Word2Vec-ESN Variant: The word2vec-ESN model variant with a reservoir of size 1000 neurons when trained and tested on all 462 sentences of corpus-462, learned to label the word in the sentences with an Accuracy(Ac), Precision(Pr), Recall(Re) and F1-Score(F1) of 97.38%, 97.48%, 92.28%, 94.64% respectively. When the model variant was tested using 10-fold cross validation we got Ac = 97.18%($\pm$0.11%), Pr = 96.86%($\pm$0.49%), Re = 91.93%($\pm$0.23%) and F1 = 94.16%($\pm$0.26%) with $IS = 1.15, SR = 0.7, LR = 0.1$. The marginal difference between the training and cross validation scores indicates that the model variant is generalizing well even on untrained data and is not overfitting. The precision, recall and f1-score for individual role is listed in table 5.2.

We also performed the simulations using Word2Vec-ESN model variant with grammatical form of sentences with localist word representation. The model produced Pr = 80.60%, Re = 84.90%, F1 = 82.31% during training and Pr = 78.04%,

Table 5.1: Mean and standard deviation of meaning and sentence error on train and test set of coprus-462 in different learning modes.

		Word2Vec-ESN		*θRARes*	
		ME	SE	ME	SE
SCL train	mean	0.541	1.515	0.123	1.207
	std	0.000	0.000	0.029	0.297
SCL test	mean	7.826	20.652	7.433	32.130
	std	1.598	2.792	0.534	1.353
SFL train	mean	0.144	0.432	0.000	0.000
	std	0.000	0.000	0.000	0.000
SFL test	mean	8.686	23.695	9.178	24.370
	std	1.265	1.170	0.574	1.192

Meaning (ME) and Sentence error (SE) in different learning modes with Word2Vec-ESN model using distributed word embeddings and *θRARes* [14] model which uses grammatical form and localist representation of words of sentences. The errors are given in percentage. SFL:Sentence Continuous learning; SFL: Sentence Final Learning; std: Standard Deviations. Simulations were done with reservoir of 1000 neurons.

Table 5.2: Training and testing classification scores for individual roles when using Word2Vec-ESN model variant.

		word2vec vectors			GF & localist vectors			
Role		Pr	Re	F1	Pr	Re	F1	Support
Agent	test	0.92	0.79	0.85	0.66	0.62	0.64	888
	train	0.94	0.80	0.86	0.71	0.68	0.69	892
Object	test	0.95	0.81	0.88	0.61	0.65	0.63	791
	train	0.96	0.81	0.88	0.67	0.69	0.68	794
Recipient	test	1.00	1.00	1.00	0.69	0.96	0.80	383
	train	1.00	1.00	1.00	0.70	0.97	0.81	384
Predicate	test	1.00	1.00	1.00	0.96	0.92	0.94	888
	train	1.00	1.00	1.00	0.98	0.94	0.96	892
No Role	test	0.97	1.00	0.99	0.97	0.96	0.97	9785
	train	0.97	1.00	0.99	0.98	0.97	0.97	9823

Comparision of training and cross validation scores for each output roles predicted by the model. Support for each role: actual number of instance, is also shown in last column. Simulation were done using 1000 reservoir neurons and parameters: SR = 0.7, IS = 1.15, LR = 0.1.

Re = 82.23%, F1 = 79.68% during cross-validation.
[Confusion matrix to be analyzed]

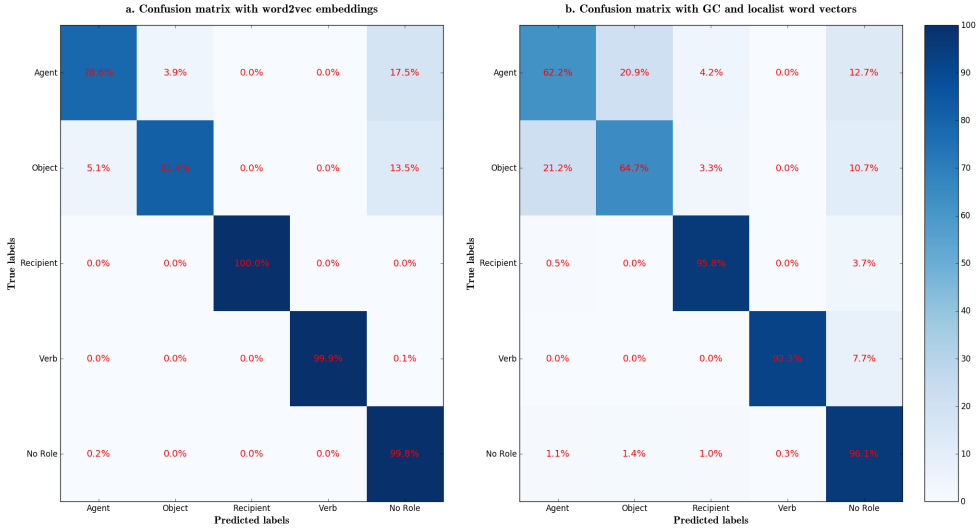

Figure 5.1: **Normalized confusion matrix with Word2Vec-ESN model variant:** The confusion matrix with true roles (in rows) and predicted roles (in columns). The top-left to bottom-right diagonal shows the percentage of words whose roles are predicted correctly. Everything other than this diagonal represents 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.

5.2.3 Experiment-3: Effect of Corpus structure

Recall that the sentences in the corpus-462 was created based on context free grammer. Thus the sentences in the corpus contains inherent grammatical structure. The model is thus possibly utilizing the underlying grammatical structure to some extend for learning and generalizing. To test this hypothesis and to demonstrate that the model is not generalizing on any other inconsitent regularity in the corpus, we removed the inherent grammatical structure from the sentence in the corpus by randomizing the word orders within the sentences [14]. Such a test will also help us to have 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 significantally less than the number of trainable parameters [14]. The Word2Vec-ESN model with reservoir of size 1000 neurons and 42 readout neurons have 42000 (42×1000) trainable parameters, whereas the model variant with reservoir size 1000 and 5 readout neuron the trainable parameters are 5000 (5×600). In

Table 5.3: Mean and standard deviation of meaning and sentence error on train and test set of coprus-462 in different learning modes.

		Word2Vec-ESN		$\theta RARes$	
		ME	SE	ME	SE
SCL train	mean	7.312	30.303	4.813	20.433
	std	0.000	0.000	0.299	1.251
SCL test	mean	69.761	99.130	74.154	99.891
	std	1.462	1.064	0.802	0.146
SFL train	mean	9.148	31.168	0.000	0.000
	std	0.000	0.000	0.000	0.000
SFL test	mean	67.548	99.347	73.391	99.913
	std	1.971	0.996	0.962	0.106

Meaning (ME) and Sentence error (SE) in different learning modes with our approach of using word2vec embeddings and Xavier's [14] approach of using grammatical construction and localist representation of words. The errors are given in percentage. For Sentence Final Learning mode (SFL): our approach (IS = 2.3, SR = 2.3, LR = 0.13) and Xavier's approach(IS = 1, SR =, LR =). For Sentence Continuous Learning mode (SCL): our approach (IS=2.5, SR=2.4, LR=0.07) and Xavier's approach(SR = 1,LR =). Simulation were done with 10 reservoir instance of 1000 neurons.

both the case the number of trainable parameters are significantally greater than our corpus size (i.e. 462 sentence). This is thus a possible situation of overfitting.

We presented the corpus with the scrambled sentences(i.e. in absence of any grammatical structure) to both Word2Vec-ESN model and its variant and performed a 10 fold cross-validation. The cross validation errors obtained in the previous experiment on the corpus 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 generalize better for the corpus with unscrambled sentences (i.e. in presence of grammatical structure). However in case of overfitting the generalization effect should not vary much both in presence and absence of grammatical structure in the corpus.

As illustrated in Table.[no], we observed that [analysis to follow....]

5.2.4 Experiment-4: Effect of Reservoir size

One of the important hyperparmeter which effects the performance of the model is the size of reservoir (i.e. number of neurons in the reservoir). Also the addition of neurons in the reserovir is computationally inexpensive, because the read-out weights (W^{out}) scales linearly with the number of neurons in the reservoir[ref?]. So, in order to determine the effect of reservoir size on the peformance of the model variant-1, we plotted the cross validation errors (see fig. 5.2) against the number of neurons in the reservoir.

It was observed that both meaning and sentence cross validatoin error reduces with increase in reservoir size but asymptotes when the reservoir size is above 1000 neurons for this corpus (i.e. corpus-462). This indicates that the model can not generalize further for this corpus with further increase in reservoir size. However the lowest errors were observed in reservoir with 2882 neurons with meaning and sentence error of 6.74%(±1.5%) and 17.39%(±4.23%) in SCL mode. In SFL model meaning error and sentence errors were observed as 8.70%(±1.62%) and 22.60%(±5.25%) respectively. It can be noticed that these lowest errors does not vary significantally when compared to errors obtained with reservoir of size 1000 neurons (see table 5.3).

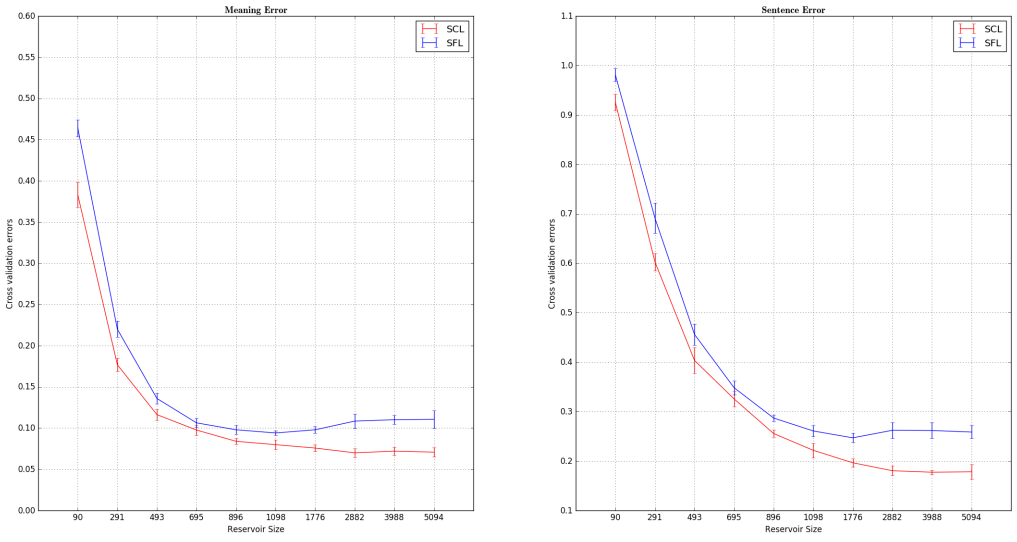

Figure 5.2: **Effect of reservoir size on cross validation errors on Model Variant-1:** Description goes here.

We also studied the effect of reservoir size on the performance of model variant-2 when using word2vec word embeddings and also when using grammatical construction of sentence along with localist word representation. In figure 5.3, it can be clearly observed that the model variant-2 performs much better with word2vec word embeddings with all reservoir size when compared to use grammatical construction of sentence with localist word vector. Even the highest F1-Score ($F1 =$

80.06%) obtained using localist representation with reservoir size 2250 is much less than that of obtained using word2vec word embeddings with reservoir of size 200 (F1 = 93.56%). Overall with increase in reservoir size the classification score also increases irrespective of the word vectors used as an input to the model. The model also asymptotes when reservoir size is 200 and 600 with word2vec and localist vectors respectively, indicating that model can not be genralized further on this corpus (i.e. corpus-462).

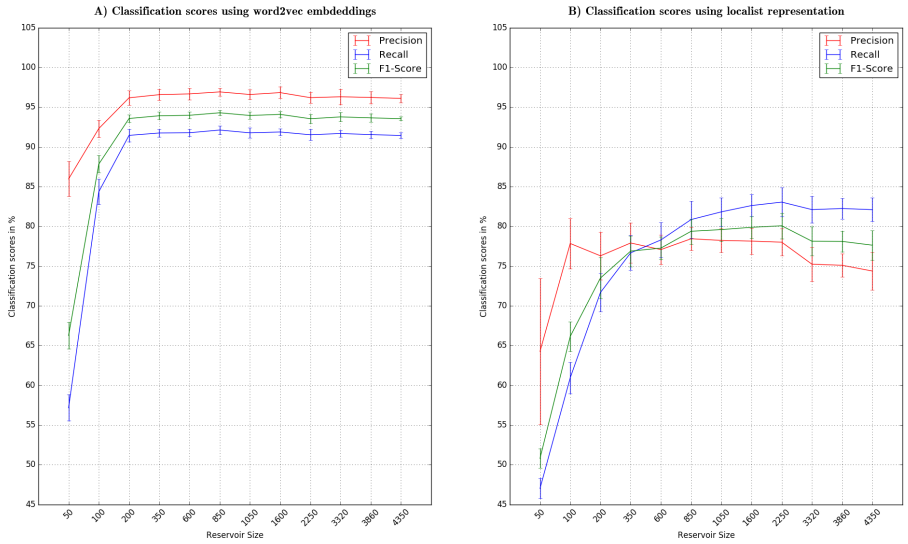

Figure 5.3: **Effect of reservoir size on classification scores of Model Varinat-2:** Description goes here.

5.2.5 Experiment-5: Effect of Corpus size

In the previous experiments we noticed that the errors rates for model variant-1 and classification scores for model variant-2 [] improved as we extended the corpus size from 45 to 462 sentences. To investigate the effect of corpus size and scaling capability of the model, we used extended corpus-90582(see section) for this experiment. As the corpus also contains 12% of ambiguous sentences which impede the learning and generalization of the model, this experiment will also validate the model's ability to process the abmbiguous sentences.

In order to study the generalization 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 corpora of 90582 sentences[42]. Each of the sub-corpora was exposed to the model and 2-fold cross validation is 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 then tested on the first half used for training previously.

Figure 5.4 shows the cross validation errors rates with respect to corpus size while using model variant-1. It can be observed that with increase in corpus size from 6% to 25%, the meaning error sharply drops from some 12.23% to 3.92% in SCL mode and from 12.10% to 4.88% in SFL mode. Similarly, the sentence error also decreases from 54.43% to 21.89% in SCL and from 55.22% to 26.17% in SFL mode. When the sub-corpora size is 50%, where the model was trained only on 25% of corpora size, the model already generalized with 17.11% sentence error and 2.98% meaning error in SCL mode and 22.32% sentence error and 4.13% meaning error in SFL mode. The more gradual slope from 50% to 100% sub-corpora size in both the learning modes for meaning and sentence error indicates the model has already generalized and further increase in corpus size wont have much effect on cross validation error.

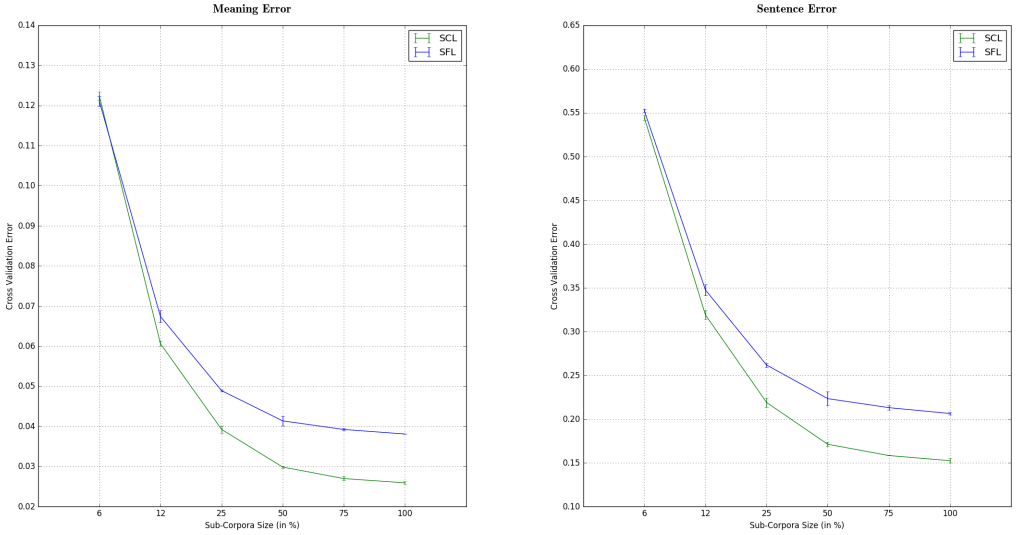

Figure 5.4: **Effect of corpus size on cross validation errors:** Description goes here.

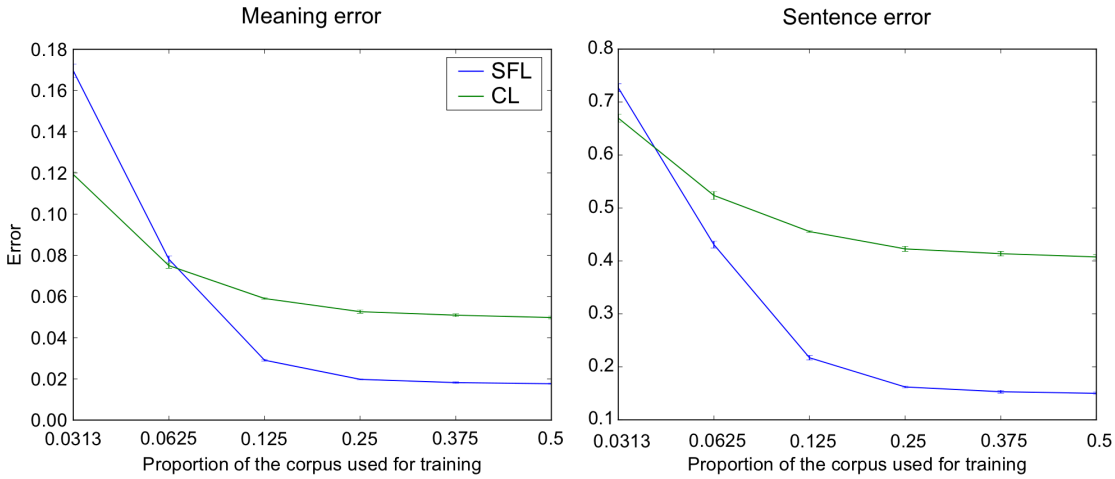

Figure 5.5: **Effect of corpus size on cross validation errors using localist word vector as reported in [ref?]:** Description goes here.

5.2.6 Experiment-6: Neural output activity of the model

In the previous experiments we observed that both the model variants generalized well and cross validation error rates dropped with increase in reservoir size. As we saw that model variants performs better when we increased corpus size from 45 sentences to 462 sentences. As corpus-45 contains sentence with uniques grammatical structure (i.e. active, passive, subject and object relative etc.) we added these 45 sentences to corpus-462 so the resultant corpus have 507 sentence (462+45) and analysed the activation produced by model variant-1 for the input sentences. We observed that while learning the model is re-analyzing the thematic roles across the time. The same behaviour was observed by xavier et. al [ref?] with sentences in grammatical constructions form and localist representation of words. However we observed that the model is making earlier predictions with word2vec embeddings. The reason for such a behavior is because the word2vec word embeddings were learned from the context words and each word vector encodes the information about neighbouring words.

We examined and plotted the four sentences with active and passive constructions studied in Hinaur et al. [ref].

1. the man gave the book to the boy.
2. the man took the ball that hit the glass.
3. the boy caught the ball that was thrown by the man.
4. the ball was pushed by the man.

Figure 5.6 shows the activations for these four different sentences across time. As all the four sentences start with "the", activation at this word is same for all the sentences. In sentences 1 and 2 with the arrival of first noun (i.e. man) the activation of noun-2 (i.e. book and ball) being an object of verb-1 is increased and confirmed with arrival of verb 'gave' and 'took' respectively.

Consider another two sentences

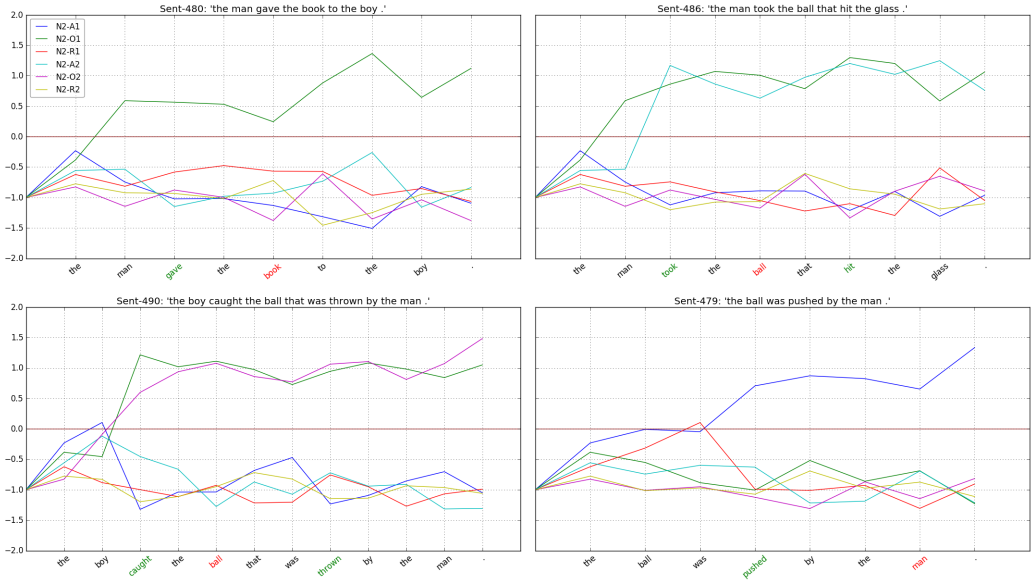

Figure 5.6: **Effect of corpus size on cross validation errors using localist word vector as reported in [ref?]:** Description goes here.

1. the dog that chased the cat ate the rat
2. the cat that the dog chased bit the man.

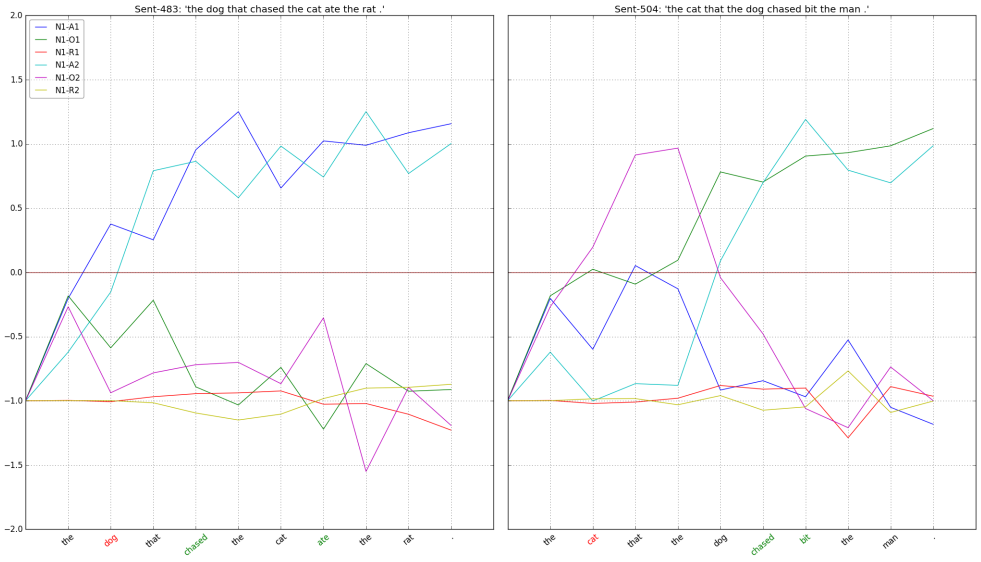

Figure 5.7: **Effect of corpus size on cross validation errors using localist word vector as reported in [ref?]:** Description goes here.

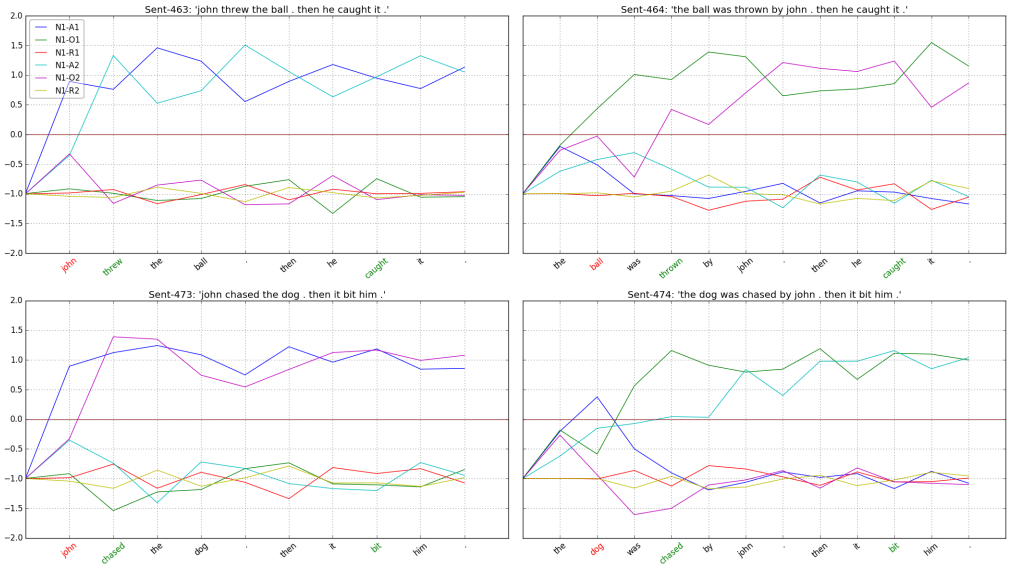

Figure 5.8: **Effect of corpus size on cross validation errors using localist word vector as reported in [ref?]:** Description goes here.

5.2.7 Experiment-7: 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 makes it easier for the model to learn and genralize on these corpus. To answer this question, in this experiment we used the corpus collected by Hinaut et al. [**?**] in a Human Robot Interaction (HRI) study of language acquisition. The sentences in the corpus were collected from the real subjects interacting with the humanoid robot (iCub). The corpus contains 373 complex instruction to perform single or double actions with temporal correlation (see action performing task in Experiments of [**?**] for more details). For example, "Point to the guitar" is a one action command whereas "Before you put the guitar on the left put the trumpet on the right" is a complex instruction with double action, where second action is specified before the first action. Thus this data is complex enough to test the learnabilty and generalization of the model.

To test the generalization of Word2Vec-ESN model on this corpus, leave-one-out (LoO) cross validation was performed. We chose LoO, so that results can be compared with that of obtained in $\theta RARes$ model of Xavier et al. [**?**]. As illustrated in table 5.4, it can be observed that while using word2vec word embeddings over grammatical form and localist word vectors, of the input sentence, error improved by 26.31%, 17.97% sentence error with 500 and 1000 reservoir neurons respectively. It can also be observed that with increase in reservoir size the sentence error also decreases. For this experiment we did not explore for the best parameters, but instead we used the optimized parameter obtained on corpus-462 in experiment 2 i.e. SR = 2.4, IS = 2.5 and LR = 0.07. This shows that previously learned model parameters are capably roboust enough to generalize on new corpus.

Table 5.4: Generalization error in sentence continuous learning mode for corpus-373.

Reservoir	Sentence 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 errors (in %) obtained with Word2Vec-ESN model and $\theta RARes$ in SCL mode for input sentences with reservoir of size 500 and 1000 neurons. The results reported are mean and standard deviation of 10 reservoir instances. The best error here means the count of sentence errors common in all 10 reservoir instances[15].

Chapter 6

Conclusion And Future Work

6.1 Conclusion

To be written...

6.2 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 [36]. CRFs has been the one of the most successful apporach used earlier as well for classification and sequential data labelling tasks [42, 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 [38] also increases till some point. However adding more dimensions results into reduced improvement [27]. 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 [28]. Thus the Word2Vec-ESN language model can also be investigated for multiple language acquisition [16].

Appendix A

Nomenclature

Appendix B

Additional Proofs

Appendix C

Complete Simulation Results

Bibliography

[1] Vector Representations of Words.

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

[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, 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.

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

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

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

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

[29] 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.

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

[31] 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.

[32] 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.

[33] 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`.

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

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

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

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

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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