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 to determine who did what to whom[ref?]. 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 [4].

Many successful traditional system consider SRL as a multiclass classification problem use linear classifier such as SVM to tackle the problem [2, 3]. These system were based on pre-defined feature templates derived from syntactic information obtained by parsing the training corpus. However in an analysis it was found that the use of syntatic parser certainly leads to degradtion of predictions [3]. 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. Also 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 [4].

Apart from the traditional methods, the task was also attempted with neural network models. [Collobert 2011] first attempted to build an end-to-end system without parsing 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[ref]. However to increase the model perfomance they also resorted to use syntatic features by using parse trees[ref].

Xavier et al., [1] proposed a generic neural network architecture using recurrent neural network (RNN) based on reservoir computing approaches, namely Echo State Networks (ESN) to solve TRA task. The choice of ESN over simple RNN has three major advantages. First, ESNs are capable of modeling long term dependecies in the sentence [ref].Second, while processing long sequence the gradiend parameter vanishes or explodes in simple RNNs. 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[ref]. The proposed architecutre models the language acquisition in brain and provided a robust and scalable implementation on robotics architecture(hinaut 2013,2014).

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 $'SW'$ and then the sentences are input to model sequentially, word by word across time. The output units coding the the role for each semantic word with respect to a verb is then learned by ESN while processing the sentence. Like other traditional NLP system they also treated words as discrete atomic symbols and used localist vector respresentaion as input for words. Treating each word as an dicrete symbol does not provide any realtional information to the model which exist between two words. For example if word pink and red is represented using localist representation by vectors [1,0,0] and [0,0,1] respectively then the semantic relationship between these two words are lost. Although replacing of semantic words with $'SW'$ token makes easy to train the model on small corpus as the $'SW'$ token can be replaced with different semantic words to form a sentence. Whereas on the other hand this makes it difficult for the model to learn thematic roles for ambiguous sentences (sentences with same surface form but different coded meaning). 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 to grammatical constructions described above, we hypothesis that the use of distributed word representation which can capture the semantic relation 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. [ref] widely known as Word2Vec model. Word2Vec model learn high quality, low-dimensional vector represenation encoding semantic meaning of words from a large corpus in an unsupervised way[ref-two]. The obtained vector embeddings also encodes several language regularities and patterns[] and can be seen by performing linear operation on the vector. For example, vector('king')-vector('man')+vector('woman') $\approx$ vector('queen'). Unlike other neural network models for obtainig word embeddings, training Word2Vec is computationally cheap and efficient[ref?] . See Chapter 2 for training word2vec model.

In this work, we thus propose to extend the xavier's model to obtain an end-to-end system by adding an additional word2vec unit trained on a general purpose unlabelled dataset(e.g. wikipedia) to generate word embeddings for the words of input sentences. The generated word vector by word2vec model can then be used by ESN for learning thematic roles of the input sentences. There are also 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. To this date, there is no research conducted with this combination of word2vec model and ESN.

Apart from adding an additional word2vec unit in the existing model we also propose a variant of this model. In this model variant input and output coding

for the model is 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 for the input word. For the evaluation of this model variant we used metrics (classification scores) proposed for CoNLL-2005 SRL task[ref]. See chapter 4 for more details.

Chapter 2

Basics of Word2Vec and Echo State Network

2.1 Word2Vec

Word2vec is a neural model proposed by Tomas Miklov that 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. Word2vec neural word embedding overcomes this issue and capture the semantics and syntactic information of the word in a computationally-efficient manner. For learning word embedding two neural architecture were proposed, the Continuous Bag Of Word (CBOW) and Skip-Gram. 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 the models is shown in [fig]. In the next sections we give a brief overview of CBOW model and skip-gram model. However we used skip-gram model in our work because it was proven that it produce better word-embeddings as compared to CBOW[ref main paper].

2.1.1 CBOW Model

CBOW model are three layered neural model with the training objective to predict a target word (e.g. Peter) given some context words (John gave ball to)[ref]. Figure shows the network architecture with a simplified case of one context word. The model is trained on the dataset having a vocabulary size V in an unsupervised way to achieve the objective. The hidden layer is of size N, the dimensions of 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 $w_c \in \mathbb{R}^V$ 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[ref].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 $u \in R^V$ is then calculated for all the target words in the Vocabulary, which is essentially the compatibility of a word w_i given the context word w_c.

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

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

where u_i gives the score of i^{th} word,w_i, for i = 1,2...V. $W^{'} \in \mathbb{R}^{N \times V}$ is the weight matrix between hidden and output layer. $W_i^{'}$ is the i^{th} column vector of matrix $W^{'}$. The computed scores are then converted to posterior probabilities by output neurons with softmax activation function. 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 a 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 words. So the cost functions then 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 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.

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. More specifically given a target word sequence $w_1, ...w_V$, 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+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.

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[ref].

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. The model is thus trained to distinguish the target word from k imaginary noise words w_{noise}, in the same context. Thus log probability $p(w_{t+j}|w_t)$ is 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}.

The objective function is thus optimized using stochastic gradient descent to learn the good word features.

2.1.3 Properties of Word2Vec embeddings

Although the word2vec model is simple in the architecture and learning, it produces word vector embedding which surprisingly encodes several linguistic regularities and patterns [ref -main paper]. More importantly it is astonishing because the network was not explicitly trained for these linguistic properties (see fig 5.4 2.2). The distributed word embeddings encodes semantic and syntactic properties of the words[ref word2vec 2013] as a constant vector offset between a pair of words sharing

a specific relationship. For example the word embeddings "$King - Queen \approx Man - Woman$", "$Apples - Apple \approx Cars - Car$".

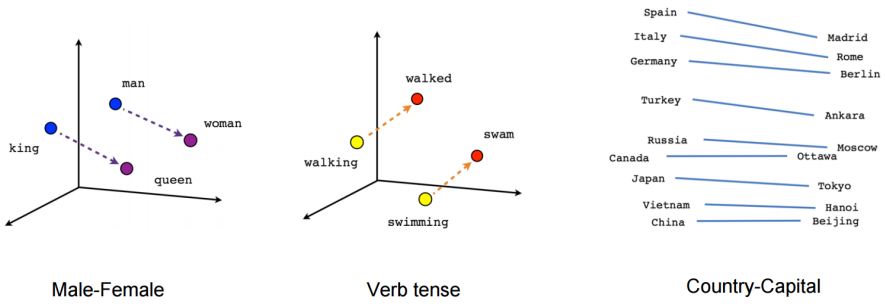

Figure 2.1: **Word2vec Semantic regularities:** Description goes here. [ESN9]

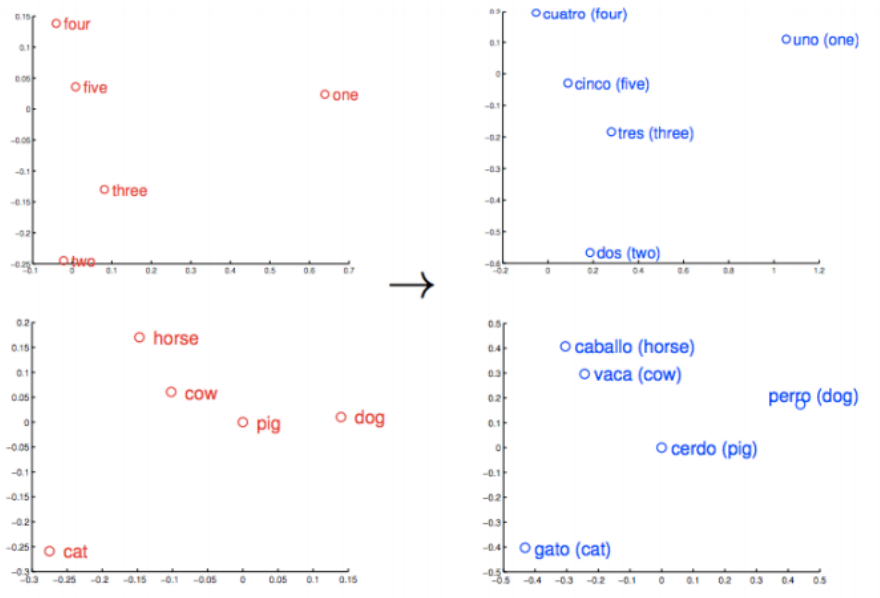

Figure 2.2: **Word2Vec Translation Property:** Description goes here [ESN9]

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 same geometrical arrangement in the respective language word embedding space. Thus it is also possible to learn linear mapping between different embedding space by vector rotation and scaling. Several other regularities can also be captured by performing basic linear operation on word-embeddings.

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 in 2001[ref] 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/sequential data [6]. 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.

2.2.1 ESN Architecture

ESN is surprisingly efficient variant for RNN training (see fig. 4.1). 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. 1 (add a foot note that we don't need these weights in our 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. For ESN approach to work the reservoir should possess the Echo State Property. Echo state property says that 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 [ref].

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 and randomly from a normal or uniform distribution i.e. most of the elements in these matrices will be zero. 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.

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

where W^{res}_{new} is the scaled reservoir weight matrix, $0 < \gamma < 1$ is the desired

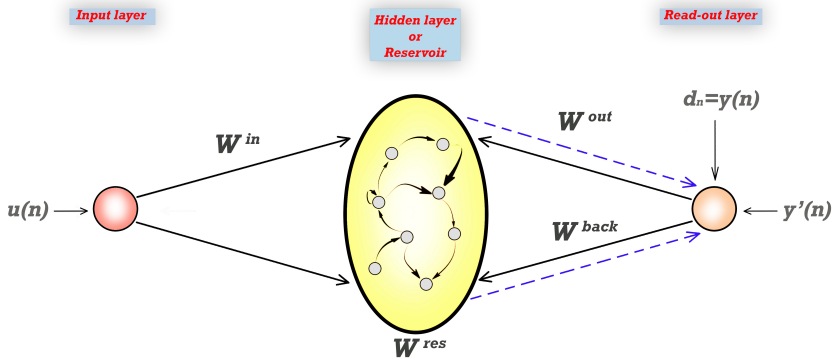

Figure 2.3: **Architecture of classical ESN:** The reservoir is the recurrent neural network with N units and initialized with random connection. The reservoir is provided input from the input u(n) and y(n) from the input and output neurons respectively. The input weights Win from input and Wback from output neurons are also randomly initialized and stays static during learning. The output weight from reservoir to output unit is learned by the network.[ESN9]

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 $\rho(W^{res}) > 1$ [ref]. 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. [practical esn]. The input weights are also scaled to regulate the non-linearity in reservoir activations. A very high input scaling let the reservoir 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 [ref].

2.2.2 Training ESN

ESN can be thought of as a supervised machine-learning problem, where for a given training input signal $u(n) \in \mathbb{R}^{N_u}$, corresponding teacher signal $y(n) \in \mathbb{R}^{N_y}$ at time-step 'n' is also provided. Here n = 1,2...,T is the discrete time step for sequence of length T. The main objective of ESN approach is to learn a model which outputs $y'(n)$, such as to minimize the error measure E(y',y) on the training data and also on the unseen data. The error measure is often the Root-Mean-Square-Error (RMSE).

$$E(y', y) = \frac{1}{N_y} \sum_{i=1}^{N_y} \sqrt{\frac{1}{T} \sum_{n=1}^{T} \left(y_i'(n) - y_i(n)\right)^2} \tag{2.2.2}$$

ESN is initialized with a reservoir of size N_x containing leaky-integrated discrete-time continuous-value neurons with tanh activation function. It is also recom-

mended to use the reservoir as big as possible and also computationally affordable[ref]. 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 [ref]. It is also advisable to proper regularization to avoid over- fitting. The reservoir weight and the input weights are also initialized. The training data *u(n)* is then input to the reservoir step by step which drives the reservoir. 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)* [6]. The reservoir activation and reservoir state update is computed using following recursive equations:

$$x'(n) = tanh\ (W^{res}x(n-1) + W^{in}.u(n)) \tag{2.2.3}$$

$$x(n) = (1-\alpha)x(n-1) + \alpha x'(n) \tag{2.2.4}$$

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, $\alpha \in (0, 1]$ is leaking rate of neurons.

The leaking rate, α , regulates the speed of reservoir update dynamics in discrete time[ref-pg].Smaller value of also induces slow reservoir dynamics thus ensuring the long short-term memory in ESN.The reservoir activation states are accumulated for an input sequence for regression with the teacher output. The linear readout weights are then learned using equations:

$$y(n) = W^{out}x(n) \tag{2.2.5}$$

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

$$Y = W^{out}X \tag{2.2.6}$$

$$W^{out} = YX^T(XX^T + \beta I)^{-1} \tag{2.2.7}$$

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 find the best parameter combination.

Chapter 3

Previous Work and Open Issues

How does humans understand the meaning of sentence from its surface form? With this research question Hinaut et al in 2013 proposed a neuro-inspired model 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(hinaut 2013,2014) 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 (Hinaut, Dominey 2013; Hinaut, Wermter 2014; Hinaut et al. 2015). The model was based on the cue competition hypothesis of Bates et al. 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 Grammatical Construction and Thematic Role Assingment

To train the ESN, the grammatical construction of sentences was used. A grammatical construction is the mapping of the surface form of a sentence (word order) to the thematic roles or meaning(see Fig. 1). The sentences were preprocessed to form an abstract representation. The abstraction marks each word of a sentence into three kind of symbols namely Function Words (FW), Semantic Words (SW) and Infrequent Words (IW). 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 etc. Infrequent words are the words which occurs and used less frequently(see table 3.1). Thus before giving a sentence as an input to ESN, all the semantic words are replaced with a unique token SW leaving functional words unchanged.

Considering each word as a discrete atomic symbol, the words are encoded using localist vector representation, where all elements except the current input are zeros (see table 3.2). The localist representation of word is input to ESN along with the

IW	SW	FW	SW	FW	FW	SW
please	put	the	ball	on	the	box

Table 3.1: Table Caption

teacher output for training across time(see fig2). Transformation of a sentence to grammatical construction as an input to ESN certainly have an advantage of training on a small dataset. The reason is that a grammatical construction can represent several sentences just by replacing SW, IW token with any open class word and infrequent words respectively.

Words	**Vectors**					
IW	1	0	0	0	0	0
SW	0	1	0	1	0	1
the	0	0	1	0	0	0
SW	0	1	0	1	0	1
on	0	0	0	0	1	0
the	0	0	1	0	0	0
SW	0	1	0	1	0	1

Table 3.2: My caption

3.1.1 Limitation of using grammatical construction

Transforming a sentence into grammatical construction 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 ambigous sentences. A sentence is said to be ambigous if it has the same grammatical construction but different coded meaning and actual meaning. The limitation can be described below in the two ambigous examples.

Example 1: Ambigous Sentneces

Consider the below three sentences having the same grammatical construction i.e. SW the SW the SW SW.

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)

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 from left to right word by word at each time step. A teacher output (possible role or label) for each semantic word is also provided 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 grammatical constructions, the ESN learning from Sentence 1 is disrupted after training with Sentence 2.This is because the second semantic word in 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 network. We believe that this problem is mainly because each input words are treated as a discrete atomic symbol, 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 probabilities activations for the second semantic (SW-2) word *green* i.e. being an *Indicator* and a *color* as shown below. It becomes difficult for the network 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 network was not able to utilize this information as words were input to the in discrete form 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 grammatical construction 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 (iii) the network is not able to resolve the ambiguity for the SW-3, which possible leads to wrong labelling of the 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 grammatical construction i.e. SW SW the SW to SW.
Training ESN on sentence 1: Training the ESN on the first sentence, the fourth semantic word *London* is trained to be as a *location*.

Testing ESN on sentence 2: Testing the ESN 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 a 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 grammatical construction as an input to the ESN have produced the promising results for the thematic role assignment task. Grammatical constructions were

input to the system using localist vector representation but the ability of training the ESN with the distributed word embedding was left unexplored. Thus we hypothesize that using a distributed word embedding (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 (Mikolov et al. 2013b). Also by using distributed embedding, performance of the network for thematic role assignment can be enhanced by avoiding disruption caused by unrelated semantic words for the sentences with the same grammatical constructions. 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 allows network to learn this dynamics from the input distributed representation of the words when trained on sentences. This semantic grouping of words is not possible in localist vector encoding as words are represented as discrete atomic symbols here.

Consider using distributed embedding of words for both the sentences in Example 2 as an input to ESN. 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 internal state of the network is not 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, the proper roles are assigned to the words although the sentence have the same surface form.

Chapter 4

Approaching Word2Vec and ESN Language Model

In this chapter we propose a model used to validate our hypothesis of using distributed embedding for improvement on thematic role assignment over grammatical construction with localist representation (see section for more details).

4.1 Proposed Model

To validate our hypothesis, we propose a Word2Vec-ESN language model for thematic role assignment task (see fig). The model is inpired from the one used by [ref] for thematic role assignment task[ref]. Word2Vec-ESN model is basically the combination of Word2Vec model and ESN (see fig). The word2vec model is used to generate distibuted embeddings of the words in input sentences. The generated embeddings are then input to ESN, which further processes these embeddings to learn and predict the thematic roles for the input sentence.

The Word2Vec model was trained using skip-gram negative sampling approach on a general purpose dataset (e.g. Wikipedia) and the domain specific dataset to learn the distributed embeddings for each word in the vocabulary(see section next). The reservoir of ESN, composed of leaky integrator neurons and sigmoids activations fuctions was 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 and hidden-to-hidden 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. The reservoir internal states for a given input sequence was accumulated across time which is then linearly combined with the desired output (thematic roles) using ridge regression to learn reservoir to readout weights. The reservoir internal states are simply the non-linear high dimensional expansion of input sequence[ref?]. Reservoir to readout weights are the only weights learned during training. The mathematical expressions for reservoir

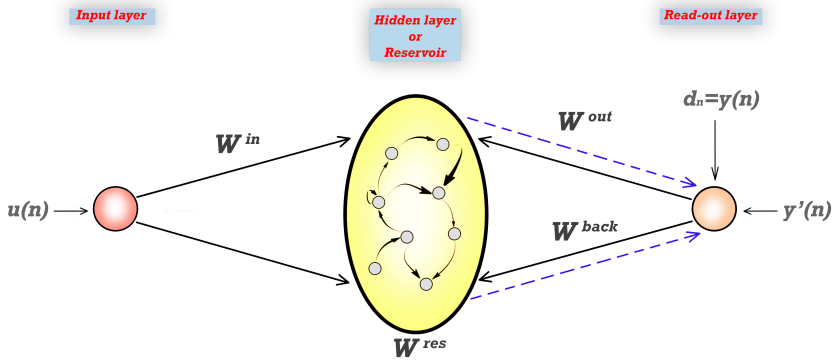

Figure 4.1: **Architecture of classical ESN:** The reservoir is the recurrent neural network with N units and initialized with random connection. The reservoir is provided input from the input u(n) and y(n) from the input and output neurons respectively. The input weights Win from input and Wback from output neurons are also randomly initialized and stays static during learning. The output weight from reservoir to output unit is learned by the network.[ESN9]

state updates and learning output weights are discussed in more details in section [ref]. The ridge regression was used rather than classical linear regression so as to avoid over-fitting and improve generalization capabilities, as it restricts the magnitude of readout weights[ref]. The learned readout weights can then later be used for generating coded meaning for the test sentences which were previously not seen by the model.

To ensure the objectivity of our findings we used two variants of the proposed models which were used to perform the experiments. Although both the variants are architecturally similar but varies in training objective and the evaluation metrics used evaluate the model performance. The number of units in input and output layer differs for each variation. Thus the input and output coding for ESN also differs depending on the training objective. The difference in variants is discussed in more details in next two subsection sections.

4.1.1 Model Variant-1

This variant of the model (see fig. 4.2) treats the thematic role assignment task as a prectition problem. The objective of the model variant is to predict the role of semantic words in a sentence with respect to a verb. To evaluate the performance of this model variant meaning error and sentence error metrics were used.(see section- Evaluation Metrics).

The training sentences are presented to the model one at time, word-by-word across time. We initially prepare two memory stacks, one for storing nouns and one for storing verbs (see fig 4.2). Word2Vec model then receives the input word and generate the vector embedding of E_v dimensions which is then input to the ESN. The input layer of ESN uses these distributed embeddings as input features for learning and predicting thematic roles of the sentences. Thus the size of input

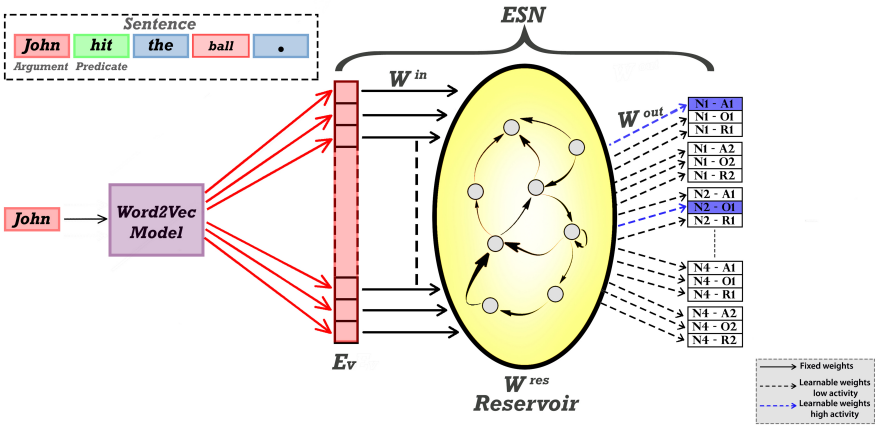

Figure 4.2: **Word2Vec-ESN Model Variant-1:** The figure shows the processing of a sentence by the model variant at time step 1. Nouns and verbs (specified in orange and green respectively) are stored in a memory stack for interpretating coded meaning. The word 'John' is input to word2vec model which 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 presented with the coded meaning of the input sentence(i.e. N1-A1: Noun-1 is Agent of verb 1, N2-O1: Noun-2 is object of verb 2). In testing, the readout units codes the predicted meaning of input sentence. The meaning: hit(John,ball,–) is decoded by mapping the thematic roles predicted by readout neurons with nouns and verbs from memory stack. Adapted from [1]

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 reservoir states are reset before the presentation of the consequent sentence. The size of readout layer depends on the maximum number of semantic words (e.g. nouns and verbs) a sentence can have in the corpus. A noun can have one of the three possible roles: Agent, Object and Recipient, with respect to a verb. For example, if the sentences in the corpus have maximum of N_n nouns and N_v verbs then the readout size would be $N_n \times N_v \times 3$ neurons; each output neuron encodes the thematic role of a semantic word (e.g. noun). The output neurons have an activation of 1 if corresponding thematic role is present for the sentence, -1 otherwise. The accumulated reservoir states are then linearly combined with readout activations to learn the reservoir-to-readout weights.

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 Mode**: In this learning mode learning takes place from the beginning of the sentence till the end of sentence or in other words the teacher labels (coded meaning) for the sentece is made availabel to model from the onset of first word in sentence. Thus, the regression is applied with teacher labels from the onset of first word in the sentence.

2. **Sentence Final Learning Mode**: In this mode the learning takes places only at the end of the sentence. 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 (see fig). As the readout neurons codes the thematic roles for individual semantic words, word level coded meaning were obtained in two steps. For every semantic word (e.g. noun) the readout activity is threshold at 0 and then the maximum of all the activations between 3 possible roles (i.e. Agent, Object, Recipient) is taken as the coded meaning of this semantic word. 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. The coded meaning of the sentence can then be interpreted from the the word level meaning by mapping the nouns and verbs from the memory stack created before giving the sentence to the model (see fig 4.2).

Evaluation Metrics

To evaluate the performance of this model variant two error measures were evaluated: the meaning error and the sentence error. 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 [ref. xavier paper]. Both the error measure are related but there is no strict corelation between them. 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.

For the analysis not all the readout neurons are considered i.e. if a sentence have only 2 semantic words then only readout neurons corresponding to these two semantic words were analyzed [ref.]. If there are more than one verb, in a sentence then each semantic word can have possible role with respect to each verb. Thus readout neurons corresponding to the mapping of semantic word and the verbs are used for analysis. For example in the sentence "John threw the ball and then he caught it". Each semantic word "John" and "ball" can have possible roles with verbs "threw" and "caught". So there are 4 possible output mappings for both the semantic words to their coded meaning.

4.1.2 Model Variant-2

This variant of the model process the sentence differently from variant-1 and consider thematic role assignment task as a classification problem. The training objective is classify the words in the sentences to one of the roles namely Verb, Agent, Object, Recipient and XX (No-role) and maximize the the classification scores (i.e. F1-score, Precision and Recall- see section-) for each role.

Training sentences are presented to the model sequentially, word by word. 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 than the 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.

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

At any time instant an argument and the predicate is input to the model where initially Word2vec model 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. The readout layer have 5 neurons each coding for a role. An output neuron have an activation 1 if the input argument have the corresponding role, -1 otherwise. The regression is performed with collected reservoir states and the readout activations to learn the reservoir-to-readout weights. Figure4.3 shows the processing of an example sentence by the model variant.

Decoding Output

Evaluation Metrics

To analyze the performance of this model variant, the confusion matrix or contigiency table (Kohavi and Provost, 1998) was build and classification scores were calculated for all possible roles: Accuracy, Precision, Recall and F1-Score. The classification scores were then averaged, weighted by support (number of true samples for each roles), to get a single real numbered scores. Weighting the average score by support takes into account the role imbalance in the dataset. The same evaluation metrics was used for CoNLL-04 and CoNLL-05 Semantic Role Labelling task[ref.].

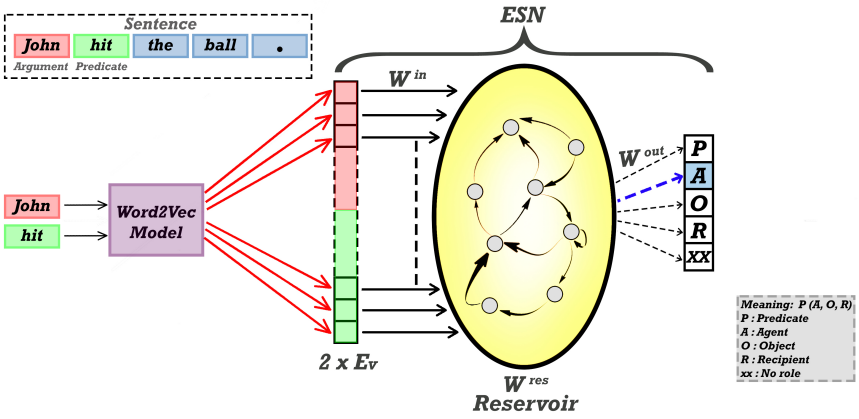

Figure 4.3: **Word2Vec-ESN Model Variant-2:** The figure shows the process of a sentence in model variant-2 at time step 1. At any instant of time an argument (current word, marked in orange) and predicate (verb,marked in green) is input to the model. Word2Vec model generates the word vectors of E_v 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 [1]

The confusion matrix describes the prediction made by the model. The rows of the matrix corresponds to the actual roles available in the dataset and the columns corresponds the predictions made by the model. The diagonal elements of this matrix represent the number of words for which the predicted role is equal to the true role, whereas all the non-diagonal elements 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 4.1.1). This meaures specifies how often the classifier is correct.

$$Accuracy = \frac{\text{number of words correctly labelled}}{\text{total number of words}} \tag{4.1.1}$$

However this measure can be distorting 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[ref.]. In our dataset we have imbalanced roles as most of words have labels "XX" (No Role) compared to other roles. 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 4.1.4). It is the measure of the accuracy of a role provided that a specific role has been predicted.

$$Precision = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{4.1.2}$$

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

$$Precision(A) = \frac{TP_A}{(TP_A + E_{BA} + E_{CA})}$$

Recall is defined as the ratio of True Positive to True Positive and False Negative. It measure how good a model is labelling the correct roles. It is also called Sensitivity or True Positive Rate.

$$Recall = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \tag{4.1.3}$$

Recall for the role Agent, in above confusion matrix will be:

$$Precision(A) = \frac{TP_A}{(TP_A + E_{BA} + E_{CA})}$$

F1-Score is the harmonic mean of precision and recall. In other words, it represents the balance between the precision and recall and is calculated as:

$$F1 = 2 \times \frac{\text{Precision} \times Recall}{\text{Precision + Recall}} \tag{4.1.4}$$

4.2 Comparision of Model Variants

4.3 Dataset and pre-processing

4.3.1 Corpus For TRA Task

For the experiments we used corpus of 462 sentence and 90582 sentences. The same corpus was used in thematic role assignment in [ref?]. The sentence in corpus-462 was generated using a context-free-grammer for English language. Each sentence in this corpus can have verbs which can take 1, 2, or 3 arguments. For example sentences, The man jump , He cut an Apple , John gives ball to Marie have 1 , 2 and 3 arguments respectively)[ref]. The sentences in the corpus have a maximum of four nouns and two verbs. A maximum of 1 relative clause is present in the sentences; verb in the relative clause could take 1 or 2 arguments (i.e., without recipient). For e.g. The dog that bit the cat chased the boy. The coded meaning for each sentence is described in order of Predicate, Agent, Object and Recipient.

Corpus-90582 consist 90582 sentences along with the coded meaning of each sentence. This corpus is redundant; multiple sentences with different grammatical

structure have the same coded meaning (see fig). In total there were only 2043 distinct coded meanings[ref] . 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) [ref]. 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., $5268 \times 2 + 184 * 3 = 11088$) of ambiguous sentences which have the similar grammatical structure but different coded meaning. Both the corpus-462 and corpus-90582 have the constructions in form:

walk giraffe ¡o¿ AP ¡/o¿ ; the giraffe walk -s . # ['the', 'X', 'X', '-s', '.'] 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 describes the order of thematic roles as appears in the raw sentence. The third part is the raw sentence with verb inflections and the fourth part is the abstract grammatical construction of sentence with Semantic words removed.

We preprocessed these constructions, to obtain the raw sentences without verb inflections. Firstly all the words are lower cased and then the verbs with inflection is modified and replaced by conjugated verb. The verb conjugation to be used depends on the inflections the verb has. For e.g The giraffe walk -s has been changed to The giraffe walks. The reason replacing verb and its inflections with verb conjugation is that we have distributed word representation which captures this syntactic relations e.g walks-walk=talks-talk. We also added additional token ¡start¿ at the 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 to obtain the word embedding of words. We chose to use Wikipedia data because we need to have vector representation of words and more words you have the better the vector representation. The Word2Vec model do not give good 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. Once the vector representation of Wikipedia data is obtained the model can be trained further on any our domain specific dataset (corpus-462 and corpus-90582) with more bias toward (by repetition of dataset during training) domain specific dataset to update the previously learned vector representations.

4.4 Word Embeddings

To get the vector representation of words we first trained a word2vec model on Wikipedia dataset. For training we used Word2vec 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 faster to train[ref?]. We used the hidden layer of with 50 units (desired dimensions of word embedding), and a context window of length +-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. For network weight update stochastic gradient, 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 for e.g. $vec(Paris) - vec(France) + vec(Germany) \approx vec(Berlin)$. But the limitation was that once the vocabulary is created from the wikipedia dataset, it was not possible add new words in the vocabulary. However there is a possibility that some of the domain specific corpus is used to train the model in extension to wikipedia corpus some words may not be present in vocabulary generated while training on wikipedia corpus. Thus we needed to update the vocabulary of the model if the word is not present in order to facilitate online training of Word2Vec model. Unfortunately neither C++ API[ref?] nor Gensim python API[ref?] implementation of Word2Vec supports vocabulary update if once created. So, adapting the idea suggested by [ref Radim and Dr, Rutu Mulkar-Mehta], we implemented the online training of word2vec by 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 few. This can be improved by repetition of new dataset several times before training the model [ref? Google group gensim].

So now when we have an online version of training word2vec model, we extend word2vec model by resuming training on 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 for further use.

Chapter 5

Experiments and Results

We performed experiments, with the previously described two variants of the proposed model. Each experiment was done with corpus 45, 462 and 90582.

5.1 Input and Ouput Coding

Model Variant-1: 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 with 1000 neurons and the readout layer with 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.

Model Variant-2: In this 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 **??**). 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 have 100 neurons where first 50 neurons encodes the vector representation of the word and remaing 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(argument) 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 the metrics(meaning and sentence) 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 genralize 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 unseen data. So in order to test the generalization capability of the model, we examine the model behavious with an extended corpus with 462 sentences(see section corpus-462) using 10-fold cross validation. Corpus-462 with 462 sentences was randomly split into 10 equally sized subsets(i.e.

each subset with 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.

Model Variant-1: We initially trained and tested the model on all the 462 sentences. The model learned the full corpus-462 with 0.54% meaning error and 1.51% sentence error in continous learning mode and 0.07% meaning error and 0.21% sentence error in final learning mode. Using the 10-fold cross validation the model generalized to 7.66%(±1.62%) meaning error and 20.26%(±2.76%) sentence error in continous learning mode with sepctral radius(SR), input scaling(IS) and leak rate(LR) of 2.4, 2.5 and 0.07 respectively. Whereas in the sentence final learning mode with $SR = 2.2$, $IS = 2.3$ and $LR = 0.13$, the optimal meaning and sentence error were observed as 8.79%(±1.24%) and 23.69%(±1.52%) respectively. The optimal parameters were identified using grid search over the parameter space.

Using word2vec vector embeddings of words as compared to localist representation we clearly see an improvement in meaning error from ...% to ...% and sentence error from ...% to ...% in SCL mode. In SFL model the meaning error reduced from ...% to ...% and sentence error from ...% to ...%.

As illustrated in Table 5.2, we observed that there is an

Model Variant-2: The model variant-2 with a reservoir of size 1000 neurons when trained and tested on all 462 sentences of corpus-462, learned to label the roles of words in the sentences with an accuracy(A),precision(P), recall(R) and F1-Score(F1) of 96.91%, 96.98%, 96.91%, 96.74% respectively. When the model varaint tested using 10-fold cross validation we got A=97.18%(±0.11%), P= 96.86%(±0.49%), R=91.93%(±0.23%) and F1=94.16%(±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 5.1.

In order to compare the performance of model variant-2 while using word2vec embeddings for words over localist vector representation, we also performed a 10-fold cross validation using localist vector representation of the words for the input sentences. The observed results indicates that the model variant-2 performs better with word2vec vectors of words than localist vector representation.

5.2.3 Experiment-3: Effect of Corpus structure

Recall that 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 generalizes well. To test this hypothesis and to demonstrate that the model is not generalizing on any inconsitent regularity in the corpus, we removed the inherent grammatical structure from the sentence in the corpus by randomizing

Table 5.1: Training and testing classification scores for individual roles

Role		*Precision*	*Recall*	*F1 Score*	*Support*
Agent	test	0.92	0.79	0.85	888
	train	0.94	0.80	0.88	892
Object	test	0.95	0.81	0.88	791
	train	0.96	0.81	0.88	794
Recipient	test	1.00	1.00	1.00	383
	train	1.00	1.00	1.00	384
Verb	test	1.00	1.00	1.00	888
	train	1.00	1.00	1.00	892
No Role	test	0.97	1.00	0.99	9785
	train	0.97	1.00	0.99	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 with SR=0.7, ISS=1.15, LR=0.1.

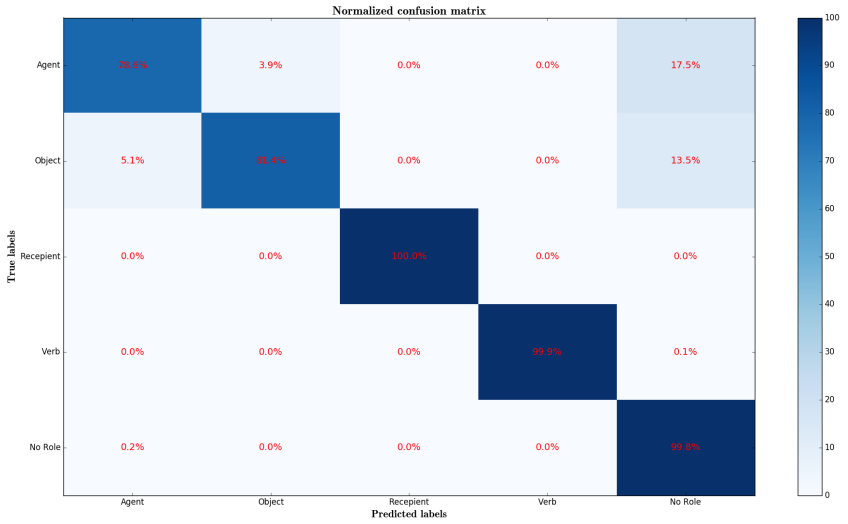

Figure 5.1: **Normalized Confusion matrix:** Description goes here.

Table 5.2: Mean and standard deviation of meaning and sentence error on train and test set of coprus-462 in different learning modes.

		Our Approach				Xavier's Approach			
		Corpus-462		462 Scrambled		Corpus-462		462 Scrambled	
		me	se	me	se	me	se	me	se
SCL train	mean	0.541	1.515	1	2	0.123	1.207	4.813	20.433
	std	0.000	0.000	1	2	0.029	0.297	0.299	1.251
SCL test	mean	7.663	20.260	69.761	99.130	7.433	32.130	74.154	99.891
	std	1.629	2.761	1.462	1.064	0.524	1.353	0.802	0.146
SFL train	mean	0.072	0.216	8.531	25.757	0.000	0.000	0.000	0.000
	std	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
SFL test	mean	8.780	23.695	69.275	99.347	9.178	24.370	73.391	99.913
	std	1.245	1.486	2.143	0.996	0.574	1.192	0.962	0.106

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

the word orders within the sentences. 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[ref?]. In the model variant-1, having reservoir of size 1000 neurons and 42 readout neurons the number of trainable parameters are 42000 (42×1000) whereas the model variant-2 with reservoir size 600 and 5 readout neuron the trainable parameters are 3000 (5×600). In both the case the number of trainable parameters are significantally greater than our corpus size (462 sentence). This is thus a possible situation of overfitting.

We presented the corpus with scrambled sentences(i.e. in absence of any grammatical structure) to both the model variants and performed a 10 fold cross-validation. The cross validation errors obtained in the previous experiment on the corpus with 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(i.e. in presence of grammatical structure) sentences. 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 ....

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 impact of reservoir size on the peformance of the model variant-1, we plotted the cross validation errors against number of neurons in the reservoir.

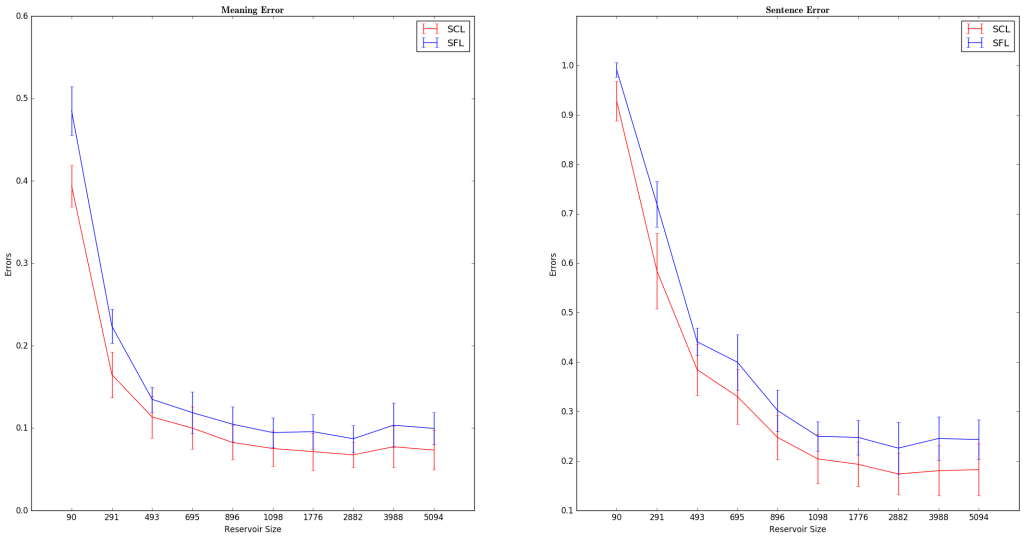

Figure 5.2: **Effect of reservoir size on cross validation errors on Model Varinat-1:** Description goes here.

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 [] 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 if the model, this experimtent will also validate the model ability to process the abmbiguous sentences.

In order to study the generalization capabilty of the model with different corpus size, we created 6 sub-corpora each containing 6%, 12%, 25%, 50%, 75%, 100% of sentences by randomly sampling from the orginal corpora with 90582 sentences[ref?]. Each of the sub-corpora is exposed to the model and 2-fold cross validation is performed to evaluate the model performance. In 2-fold cross validation the model is trained on half the subcorpora size and tested on remaining half.

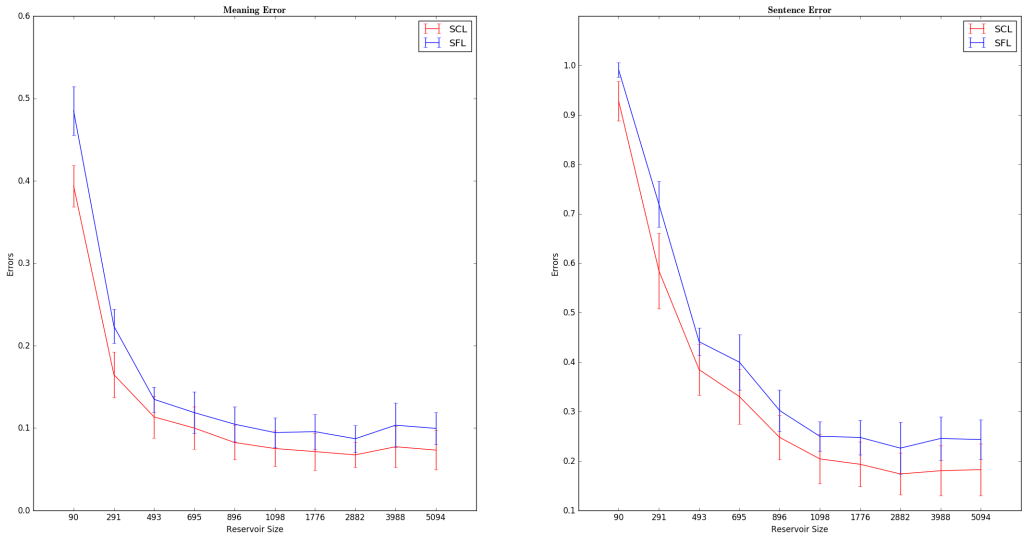

Figure 5.3: **Effect of reservoir size on cross validation errors on Model Variant-2:** Description goes here.

The second half used for testing is then used to train the model and then tested on the first half used for training previously. Figure [ref] and Figure [ref] shows the cross validation errors rates with respect to corpus size for model variant-1 and variant-2 respectively.

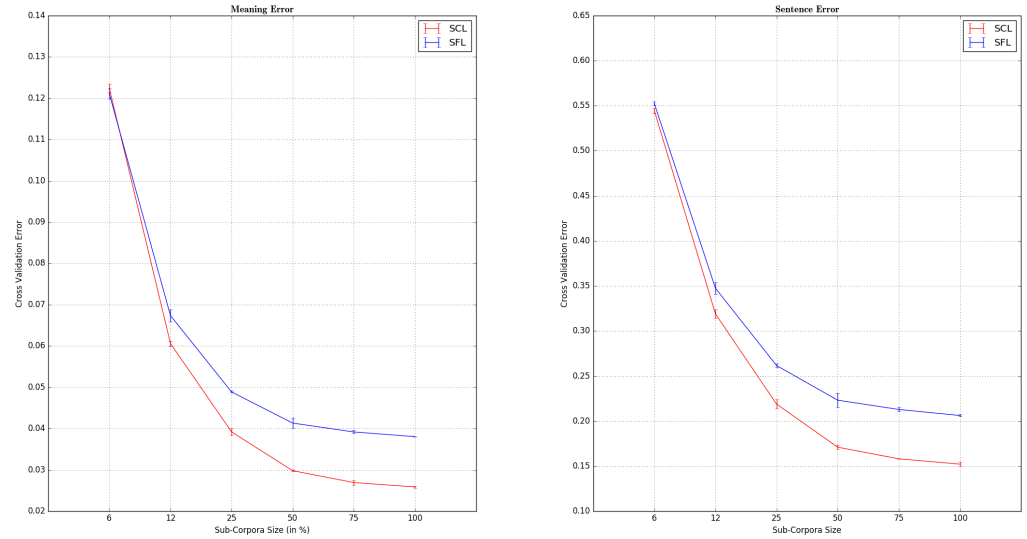

Figure 5.4: **Effect of corpus size on cross validation errors:** Description goes here.

[ESN9]

5.2.6 Experiment-6: Online reanalysis of sentences

Chapter 6

Conclusion And Future Work

Appendix A

Nomenclature

Appendix B

Additional Proofs

Appendix C

Complete Simulation Results

Bibliography

[1] 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.

[2] 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.

[3] 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.

[4] 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