DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master Thesis in Informatics

Deep Learning Methods for the Extraction of Relations in Natural Language Text

Pankaj Gupta

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master Thesis in Informatics

Deep Learning Methods for the Extraction of Relations in Natural Language Text

Deep-Learning-Methoden für die Extrahierung von Relationen in natürlichsprachlichem Text

Author:	Pankaj Gupta
Supervisor:	Prof. Dr. Thomas Runkler, TUM, Siemens
Advisor:	Heike Adel, LMU
	Dr. Bernt Andrassy, Siemens
	Dr. Hans-Georg Zimmermann, Siemens
	Prof. Hinrich Schütze, LMU
Submission Date:	Nov 05, 2015

I confirm that this Master Thesis in Informatics is my own work and I have documented all sources and material used.

Munich, Nov 05, 2015 Pankaj Gupta

Acknowledgments

This Master thesis would not be possible without the support of many people. First and foremost, I would like to thank my thesis supervisors Prof. Dr. Thomas Runkler and Prof. Hinrich Schütze, for providing me the opportunity to work under their supervision. You both helped me in formulating this thesis topic and providing me with a perfect balance of guidance and freedom. Thank you for your trust in me and your continuous help. I also thank Dr. Hans-Georg Zimmermann for your informative technical sessions as well as handful of tricks to train the models. I really like your pragmatic approach to complex problems. Your technical sessions provided me with lot of insight. I would like to thank you all for your proposal discussion and review meetings that defined my project with more impact. Thank you also for reading the thesis draft and providing valuable review comments. I am lucky enough to have such caring supervisors.

I would also like to thank Prof. Schütze to introduce me with Heike Adel, his current PhD student. Heike, you have been a great advisor and I really enjoyed working with you. I would like to thank you for your continuous guidance and support, providing me the data and specially, for your quick responses to my queries and in particular to your email responses even on weekends. I would also thank you for allowing me to participate in the 2015 TAC KBP SF research challenge in collaboration with your CIS team at LMU. It was challenging time during our submission to this research competition. I thank you to keep me motivated and advised to finish a number of experiments within the due time. I would thank you for integrating my thesis work in your slot filling system. Thank you for reading multiple drafts of this thesis proposal and report. Your valuable review comments helped in refining my thesis report. I admire your ability to see the nuances in everything. I feel honored being your first Master thesis student and I hope we continue our collaboration in future too. I would also like to thank Dr. Ngoc Thang Vu for your advice and discussions during our collaborative work for the conference papers. Thank you for being my technical and non-technical advisor for all the time we had together. You are valuable source of ideas and comments, integrated with my work. Thang and Heike, you both are amazing personalities, very helpful, caring, friendly and true by heart. You always treat me as a friend that made my time more fun and productive.

Dr. Bernt Andrassy, you are one of the pillars of this thesis work. I am grateful to you that you provided me the platform and remained a concrete bridge between Siemens and CIS LMU. It is due to you that I could collaborate with great minds and special people at Siemens and CIS LMU. You provided me the required freedom to research and always stood up to help me in any situation. You have been a supportive advisor as well as an expert manager too. It has been a long healthy association for last 1.5 years and you have been very quick in resolving all my administrative issues. Thank you for all the care, motivation and valuable comments that really boost my performance. I would also take the opportunity to thank Dr. Ulli Waltinger for his time to provide his constructive inputs during the technical reviews.

I thank you Prof. Schütze and Heike to provide me the computation power at CIS LMU. I would also thank Dr. Denis Krompass and Dr. Sigurd Spieckermann at Siemens to setup and provide me infrastructures, especially GPUs, required for my thesis work. Without you all, it was not possible to finish the countless experiments within the due time.

I would like to thank God to bless me such a talented and supportive wife, Soumya Dubey. You are a special person to me. Without your constant support, care and motivation, this thesis work would be difficult.

I acknowledge with gratitude, financial assistance from Siemens, especially Knowledge Modeling Retrieval (KMR) department, to support my thesis work. I also thank Mr. Michal Skubacz who has been so supportive throughout my thesis work at Siemens. Thank you all my colleagues, including Sanjeev Kumar Karn at Siemens, for supporting and joining me in the technical discussions.

It is due to the friendly and supportive environment in Siemens as well as in CIS LMU that I was lucky enough to find so many great people to work with. I really enjoyed my collaborations with you all.

Abstract

With much human knowledge residing in the text documents and the amount of unstructured information growing constantly on the web, as a result the problem of extracting desired information is getting difficult by the day. There is a need of intelligent systems to automatically extract meaningful information from the unrestricted natural language text into structured forms. This means that knowledge bases (KBs) must be extracted and synthesized from natural language text. To alleviate this problem, various approaches to Information Extraction (IE) are currently being used to develop knowledge base population (KBP) methods that address the afore mentioned characteristics.

The goal of this thesis work is to develop learning models that can deal with unstructured data to extract relevant information, by automatic learning the necessary intermediate representations of the linguistic unit, without using any costly natural language processing (NLP) tools and handcrafted feature engineering. In order to achieve the goal, we focus on semantic relation classification that plays a key in various complex NLP tasks such as automatic knowledge base construction, opinion mining, sentiment analysis, question answering and slot filling. Relation classification task assigns a relation label, from the pre-defined classes, to a pair of nominals marked in the sentence. Several conventional approaches have been applied to this task that rely on the human designed patterns and use costly extra NLP tools, therefore to handle the difficulties and to generalize the methods in order to span various application domains without using any linguistic features from the costly NLP tools, we focus on relation classification between the mark nominals using the information in the same sentence by automatic feature learning via recurrent neural networks. We also integrate our relation classification component in the slot filling system.

The chapter 2 is an introductory chapter that introduces the general recurrent neural networks (RNNs) and discusses the related works in relation classification and slot filling. It explains the difference in sequence and relation classification tasks, and how we approach the relation classification using standard RNNs and address the shortcomings in the existing methods, which we further investigate in the subsequent chapters of this thesis work.

Further, the chapter 3 explores the model training and key features investigated in the framework of recurrent neural architectures. It also talks about the various problems during RNN optimization and the handful tricks used while RNN model training.

The chapter 4 turns to the designs of recurrent neural network for relation classification task. The chapter is the core of our thesis work, where we majorly explore and emphasize the need of bi-directionality for the relation classification task. Here, we propose the various RNN architectures, weight sharing concepts and demonstrate the recursive composition and propagation of patterns in RNNs. We also introduce ranking in RNNs and further combine existing CNN and proposed RNN models. These contribute to the major outcomes of this thesis work.

Relation classification is one of the components of slot filling task. In chapter 5, we briefly give an overview of slot filling task and talk about the introduction of RNNs into the slot filling pipeline. The key experiments and results are included in chapters 5 and 6. We present the *preliminary* results of the 2015 Text Analysis Conference Knowledge Base Population Slot Filling (TAC KBP SF) research challenge, along with evaluation on 2014 slot filling dataset. Here, we observe that RNN integration into the existing slot filling system (developed by CIS LMU) contributes to system performance leading to improving scores in the research challenge, where we participated with CIS LMU team. Combining the existing CNN and our proposed RNNs leads to state-of-the-art performance on SemEval10 dataset, which we further report to AAAI-16 conference. In these chapters, we also present an interesting analysis to understand the nature of CNN and RNN models and their performance with various sentence lengths. We extract the most representative N-gram patterns detected by the proposed RNNs for both slot filling and SemEval10 datasets. The semantic meaning accumulation nature of RNNs is also demonstrated through example sentences, that proves the effective nature of recurrent neural network in relation learning.

Contents

1 Introduction

The connection is indispensable to the expression of thought.
Without the connection, we would not be able to express any continuous thought,
and, we could only list a succession of images and ideas isolated from
each other and without any link between them. **[Tesnière, 1959]**

1.1 Overview

With the growth of Internet and plethora of documents produced by the humanity, the amount of unstructured text data is huge, so is the need of intelligent systems to automatically extract knowledge from it. The goal of the thesis work is to develop learning models that can deal with these unstructured data to extract useful information, by automatic learning the necessary intermediate representations of the linguistic unit, without using any costly natural language processing (NLP) tools and handcrafted feature engineering, and that can span various application domains. There are number of natural language task such as information extraction from large unstructured data, question answering, sentiment analysis, ontology learning, slot filling, opinion mining, etc addressed by several learning methods. We focus on the semantic relation classification that plays a key role in various complex NLP scenarios.

The aim of semantic relation classification is to assign a relation label, from the pre-defined classes, between the marked nominals in the given sentence. Semantic meaning of a relation is formed in the context of the two target nominals or relation arguments, including the word sequence between them and a window of proceeding and following words. Also, a relation has directionality i.e. the order of the context words does matter. Therefore, relation learning is essentially a task of temporal long-distance pattern learning. For instance, in the following two sentences with relation *cause-effect*,

$S1$: <e1>Conflict</e1> had caused the <e2>collapse</e2> of the Somali Republic
$\rightarrow$ cause-effect(e1, e2).
$S2$: The <e1>tides</e1> are caused by the <e2>gravitational force</e2> of the Moon
$\rightarrow$ cause-effect(e2, e1).

Conflict and *collapse* are the relation arguments for $S1$, while *tides* and *gravitational force* for $S2$. One can observe that the order of context words ('caused the' and 'caused by the') is significant in assigning the relation label to each sentence.

We address the major shortcomings of the standard approaches to deal with this task. The conventional approach to relation classification are based on pattern matching and it relies on the human-designed patterns. It requires an expert experience and is time consuming. Additionally, other approaches rely on extra NLP tools to derive linguistic features and do not generalize through various application domains. In standard NLP methods, such as *bag-of-words* approach, the order of words is ignored, which leads to obvious problem in capturing the order and directions in the context. Hence, we investigate the recurrent neural networks (RNNs), without using any linguistic features, that can accumulate the semantic meaning of sentence by processing it word-by-word at each time-step.

The TAC KBP slot filling task addresses the challenge of gathering information about entities (persons or organizations) from a large amount of unstructured text data. Relation classification is also one of the component of such automatic knowledge base construction. We investigate the performance of RNNs for relation classification in slot filling task. The TAC (Text Analysis Conference) KBP (Knowledge Base Population) Slot Filling (SF) track, organized by NIST, is a research competition where the task is to extract the values of specified attributes (i.e. slots) for a given entity from large collections of unstructured natural language texts. Examples of slots include age, birthplace, and spouse for a person or founder, members, and parents for organizations. We develop and integrate RNNs into the candidate slot filler classification component of the existing slot filling pipeline (by CIS LMU) and analyze the impact of recurrent learning method in populating knowledge bases (KBs) from unstructured text.

1.2 Problem statement

To handle the difficulties in the pattern design, the lack of annotated data, costly handcrafted feature engineering and to generalize the methods in order to span various application domains without relying on extra NLP tools, we focus on relation classification between nominals tagged in the same sentence by the automatic feature learning using recurrent neural networks.

The Convolution neural network is one of the popular deep learning models investigated for relation classification, but we address the potential problem of CNN model in learning long distance patterns. We demonstrate the semantic meaning accumulation nature of RNNs that captures the global patterns over the complete word-sequence, in contrast to CNN models which learn only local patterns. As mentioned in section 1.1, the directionality (i.e. the order of words) is also a significant aspect in assigning the relation label to a sentence. Therefore, we address the bi-directionality of RNN models and also investigate the joint training of forward and backward networks. We propose the novel RNN architecture, named as Connectionist bi-directional RNN (C-BRNN) that tackles the difficulty of CNN models in learning long-distance patterns and the lack of capability to learn temporal features in an effective way. Several variants of bi-directional recurrent neural networks are evaluated for relation classification task, in order to incorporate the complete information from word sequence, both proceeding and following at each time-step.

SemEval10 Task 8 dataset has 19 classes, including an *artificial* class that groups the sentences which do not belong to any of the 18 *natural* classes. Since a common pattern is difficult to learn for this *artificial* class and ultimately it introduces noise into the data. Therefore, there is a need to handle such an *artificial* class and focus learning on the *natural* classes. Here, we introduced a ranking layer into C-BRNN model (R-RNN), instead of softmax to handle such *artificial* class that allows the pattern learning to focus on non-artificial classes by reducing the impact of artificial class and to learn to maximize the difference to the best competitive labels.

Even, it is difficult to capture the long term patterns using RNNs due to optimization problems such as vanishing and exploding gradients. In order to handle these long distance patterns, the multi-step RNN is explored in the framework of C-BRNN model. While, for short term patterns, N-gram input instead of single input word vector to RNN, helps in detecting the most representative N-grams.

The proposed RNN architectures are evaluated on SemEval10 Task 8 data set and also integrated into the slot filling system for the 2015 KBP TAC SF research challenge submission, where we tackle the longer contexts and multiple entities/nominals (slots and fillers) in the same sentence. The combination of RNN and pre-existing CNN leads to advantage from the two different types of sentence representation. It is observed that RNN is superior to CNN performance for large context lengths.

We have the following assumptions for relation classification on SemEval10 dataset:

1. Entities are given : No entity identification and no entity disambiguation
2. Entities in the same sentence, no co-reference.

1.3 Research questions

We deal with the following research questions in our thesis work :

- Need of the bi-directionality in the context of relation classification ?
- Joint training of forward and backward networks in bi-directional RNNs ?
- Impact of weight matrices sharing in forward and backward network ?
- Semantic composition of phrases via recursive learning ?
- Handle noisy classes during model training ?
- Capture long-term dependency patterns from RNN ?
- Demonstrate the semantic accumulation nature of RNN by sequential learning ?
- Illustrate the performance of CNN and RNN based on their sentence representation and it's length ?
- Detect the most representative N-grams in the word sequence, that contributes to relation prediction ?

2 Related work

2.1 Recurrent neural networks for sequence classification

Neural networks have been extensively studied for language modeling, which computes the probability of the occurrence of a number of words in a particular sequences. The first deep neural network architecture model for natural language processing was proposed by *Neural Language Model* [**Bengio2003**], who used feedforward neural networks with fixed length contexts and captured this contexts via learning a distributed representation of words. A major difficulty of the *Neural Language Model* is that a feedforward network uses fixed length context that needs to be specified before training. This means that only a finite window of previous words would be considered for conditioning the language model neural, when predicting the next one. It is well known that humans can exploit longer context with great success. To tackle the difficulties of this model, recurrent neural network based language model [**Mikolov2010**] are capable of encoding temporal information implicitly for contexts with arbitrary lengths by using recurrent connections and information can cycle inside these networks for arbitrary long time. We use recurrent networks to model word sequences i.e. sentences, to capture the longer contexts (preceding and following) at each time step word-by-word.

We first discuss in detail the conventional RNN models for sequence modeling, and then we investigate the capability of uni- and bi-directional RNNs in capturing the semantic meaning of word sequence for relation classification task, where the significant difference of our relation classification models is that there are no predicting targets at the each time step, and the supervision (relation label) is only available at the end of the sequence, i.e. sentence-level relation classification. Since we have a single relation label for all the relation arguments existing in the sentence and our task is to predict the relation between the relation arguments (nominal) using the context from the sentences, where these relation arguments exist. Therefore, for each sentence with these relation arguments, we end up with a single label, instead of a label for each word in the sentence as in sequence classification task.

2.1.1 Standard uni-directional RNNs

The Uni-directional RNNs look into the past words to predict the next word in the sequence in language modeling paradigm. While in relation classification framework, the context accumulated from the preceding words is propagated from each time step to the end of the word sequence, where the predicting target (relation label) is available and supervision is performed. We first describe the two RNN variants for sequence classification [**Mensil2013**] : Elman-type [**Elman1990**] and Jordan-type [**Jordan1986**] RNN architectures.

Standard Elman-type RNN for sequence classification

In standard Elman-type recurrent networks [**Elman1990**] Fig 2.1, the output of the hidden layer at time $t-1$ is kept and fed back to the hidden layer at time t, along with the raw input x_t for the time t. At each time step, the input is propagated in a standard feed-forward fashion, and then a parameter updating rule is applied, taking into account the influence of past states through the recurrent connections. In this way, the context nodes always maintain a copy of the previous values of the hidden nodes, since these propagate through the recurrent connections from time $t-1$ before the updating rule is applied at time t. Thus, the network can maintain and learn a sort of state summarizing past inputs, allowing it to perform tasks such as sequence-prediction that are beyond the power of a standard feed-forward neural network.

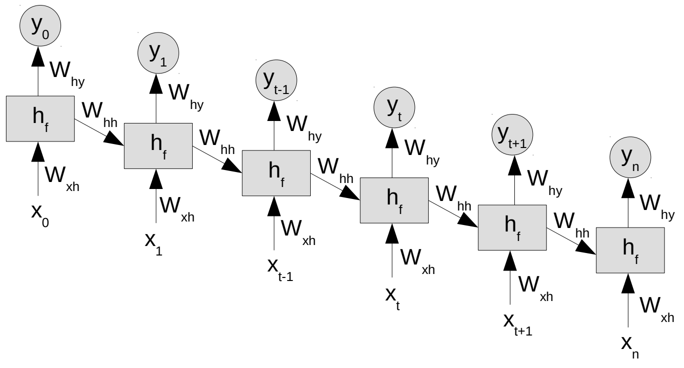

Figure 2.1: Standard Elman-type Uni-directional RNN for sequence classification.

The dynamics of the Elman-type RNN (Fig 2.1) can be represented mathematically:

$$h_{f_t} = f(W_{xh}x_t + W_{hh}h_{f_{t-1}}) \tag{2.1}$$

$$y_t = g(W_{hy}h_{f_t}) \tag{2.2}$$

where;

1. $x_0, x_1, x_2, ..., x_n$: the input sequence, i.e. *word vectors.*
2. x_t : the input word vector at time t; $x_t \epsilon R^d$.
3. h_{f_t} : the hidden layer (context or state) from all past words to x_t in word sequence; output of the non-linear function at the time, t; $h_{f_t} \epsilon R^{D_h}$.
4. f : the activation or non-linearity function at the hidden layer, h_f. The possible choices are *sigmoid, tanh, rectified linear units (ReLu), cappedReLu*.
5. y_t : the prediction output i.e. sequence label at each time step; has dimensionality the number of classes, C.
6. g : the softmax function at the output layer; where $g(z_m) = \frac{\exp(z_m)}{\sum_k \exp(z_k)}$
7. W_{xh} : the weights matrix used to condition the input word vector, x_t; $W_{xh} \epsilon R^{d \times D_h}$.
8. W_{hh} : the weights matrix used to condition the output of the previous time-step, $t-1$; $W_{hh} \epsilon R^{D_h \times D_h}$.
9. W_{hy} : the output weight matrix ; $W_{hy} \epsilon R^{D_h \times C}$.

Standard Jordan-type RNN for sequence classification

Jordan-type RNNs are similar to Elman-type networks, except the difference in the hidden layer input. The hidden layer at time t is fed from the output layer y_{t-1}, instead of the hidden layer unit at time $t-1$, as in Elman-type networks.

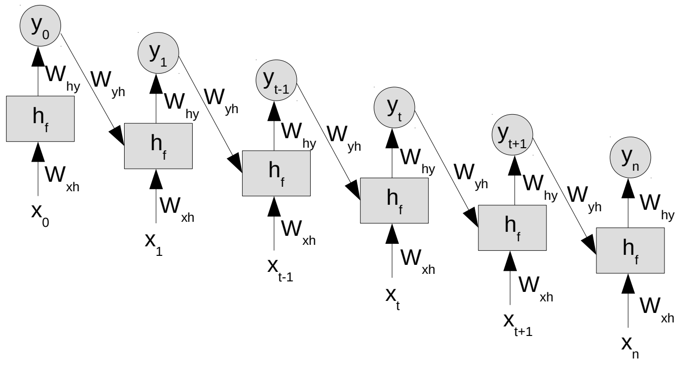

Figure 2.2: Standard Jordan-type Uni-directional RNN for sequence classification.

The only difference in hidden layer input (Fig 2.2) is mathematically given by :

$$h_{f_t} = f(W_{xh}x_t + W_{hy}y_{f_{t-1}}) \tag{2.3}$$

where; W_{hy} : the weights matrix between the output unit and hidden layer unit; $W_{hy} \epsilon R^{C \times D_h}$.

We shall investigate the Uni-directional RNN architectures for relation classification late in the following chapters.

2.1.2 Standard bi-directional RNNs

To overcome the representational power of the Elman-style uni-directional RNN, and to use all the available input information in the past and future of a specific time step, [**Paliwal1997**] proposed to use two separate networks, each for time direction and then merged the results. However, there are various ways ([**Jacobs1995**]) to merge the results from the forward and backward networks. Another simple way to work around the uni-directional RNN problem is to include a fixed-size future context around a single input vector (token). However, this approach requires tuning the context size, and ignores future information from outside of the context window.

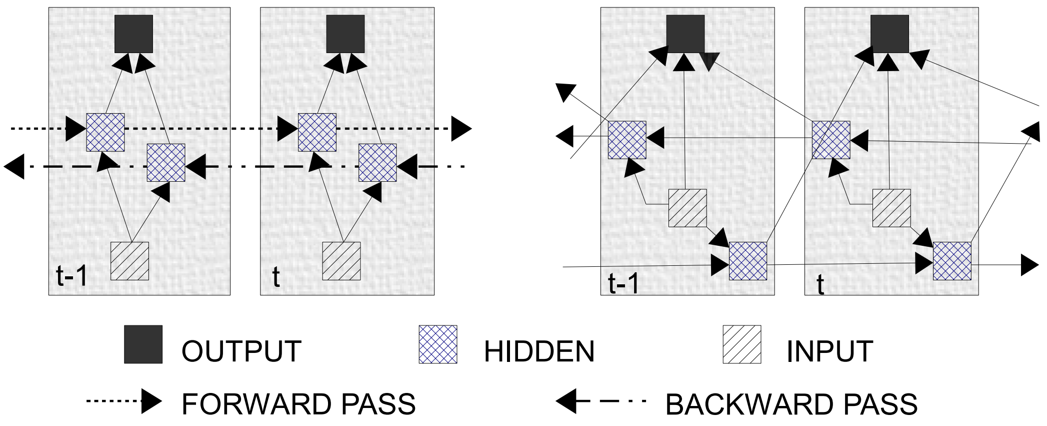

Figure 2.3: ((Left) General architecture of BRNNs [**Paliwal1997**]. (Right) BRNNs proposed by [**Paliwal1997**].

As described in Fig 2.3, [**Paliwal1997**] proposed connecting the forward and backward states to the current output states, they are connected to the next and previous output states, respectively, and the inputs are directly connected to the outputs. They used Bayes rule $p(x,y) = p(x|y)p(y)$ to estimate the conditional posterior probability $Pr(c_1, c_2, ...c_T|x_1^T)$ of a sequence of all classes from $t = 1$ to $t = T$ instead of

$Pr(C_t = k|x_1^T)$, given the sequence of input vectors. The sequence posterior probability is decomposed as:

$$Pr(c_1, c_2, ...c_T|x_1^T) = \underbrace{\prod_{t=1}^{T} Pr(c_t|c_{t+1}, c_{t+2}, ...c_T, x_1^T)}_{backward posterior probability} = \underbrace{\prod_{t=1}^{T} Pr(c_t|c_1, c_2, ...c_{t-1}, x_1^T)}_{forward posterior probability} \tag{2.4}$$

The probability term within the product is the conditional probability of an output class given all the input to the right- and left-hand side plus the class sequence on one side of the currently evaluated input vector. They trained two separate forward and backward networks to estimate conditional probabilities, which then combined by using the formulas above to estimate the full conditional probability of the sequence. Mathematical representation of Fig 2.3,

$$h_{f_t} = f(W_{f_{xh}} x_t + W_f h_{f_{t-1}}) \tag{2.5}$$

$$h_{b_t} = f(W_{b_{xh}} x_t + W_b h_{b_{t+1}}) \tag{2.6}$$

$$y_t = g(W_{f_{hy}} h_{f_t} + W_{b_{hy}} h_{b_t}) \tag{2.7}$$

where; W_f and $W_{f_{xh}}$ are forward weight matrices; W_b and $W_{b_{xh}}$ are backward weight matrices; $W_{f_{hy}}$ and $W_{b_{hy}}$ are output matrices for output y_t for input x_t at time step, t; h_{f_t} and h_{b_t} are hidden units for forward and backward pass respectively. When a decision is made on an input vector, the two intermediate representations or summaries, h_{f_t} and h_{b_t}, of the past and future are utilized and combined at each time step, t.

Note that the forward and backward parts of the network are independent of each other until the output layer when they are combined. This means that during training, after back-propagating the error terms from the output layer to the forward and backward hidden layers, the two parts can be thought of as separate, and each trained with the classical back-propagation through time [**Werbos1990**]. Here, in our work, we shall also address the joint training of forward and backward network in BRNNs.

There are other variants of BRNN for sequence classification, based on the different types of combining the forward and backward network at the output. Fig 2.4 and Fig 2.5 defines two ways of combining forward and backward network, for the task of sequence classification. In both networks, additional hidden (h_{bi}) representation are generated by combining forward and backward histories, and then output units (y_t) are attached to these combined hidden representations at each time step. While, in Fig 2.3 (right), the output units from forward and backward units are combined using Bayes

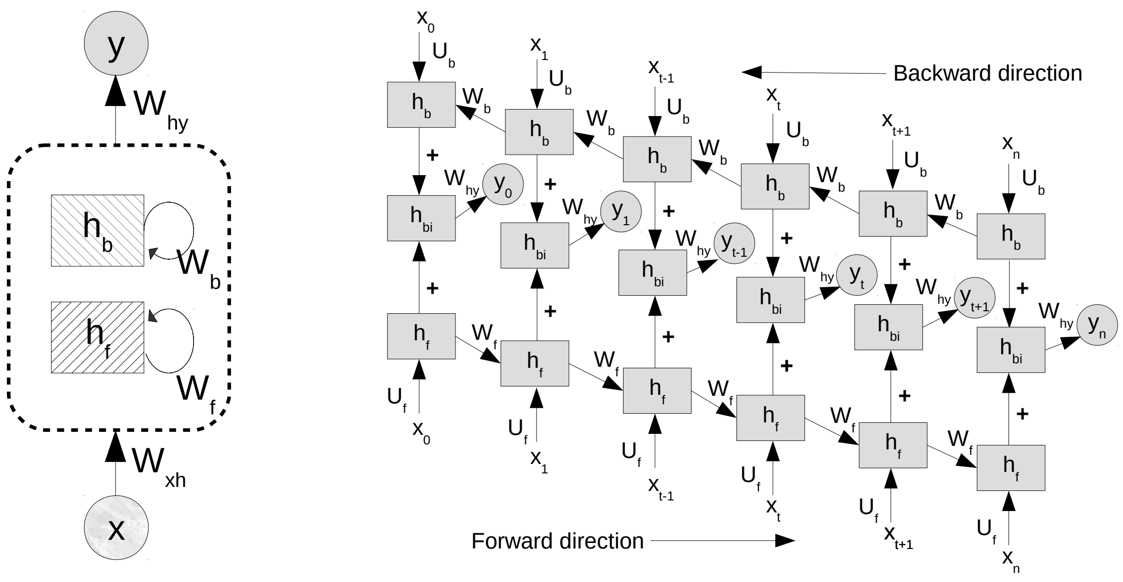

Figure 2.4: Standard Bi-directional RNN (variant 1) with independent weights for sequence classification.

rule, instead of combining the forward-backward hidden units to perform supervision. Fig 2.4 and Fig 2.5 are different in the way that the order of hidden representations for each input are combined. In the first case Fig 2.4, the hidden representations of the same input (e.g. for x_0) at each time step are combined from forward and backward network; while in other case Fig 2.5 , the hidden units representation of the revered input sequence from backward pass are combined with the forward pass hidden units (e.g. for the first, x_0 and last input x_n in sequence). [**Mesnil2013**] investigated such recurrent neural network architectures, Fig 2.4, in Jordan-type style with weights to combine forward and backward network, for Spoken Language Understanding.

Mathematical representations are given as:
The forward and backward hidden representations in Fig 2.4 and Fig 2.5:

$$h_{f_t} = f(W_{f_{xh}} x_t + W_f h_{f_{t-1}}) \tag{2.8}$$

$$h_{b_t} = f(W_{b_{xh}} x_t + W_b h_{b_{t+1}}) \tag{2.9}$$

The bi-direction hidden units in Fig 2.4 and Fig 2.5, respectively:

$$h_{bi_t} = h_{f_t} + h_{b_t} \tag{2.10}$$

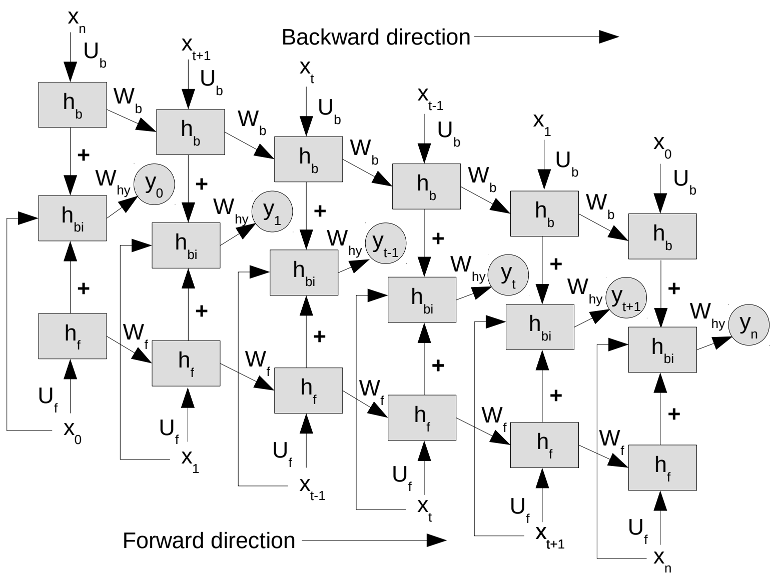
Backward direction
Forward direction
x_n U_b h_b W_b h_{bi} W_{hy} y_0
h_f W_f U_f x_0
x_{t+1} x_t x_{t-1} x_1 x_0
y_1 y_{t-1} y_t y_{t+1} y_n
x_1 x_{t-1} x_t x_{t+1} x_n
+

Figure 2.5: Standard Bi-directional RNN (variant 2) with independent weights for sequence classification.

$$h_{bi_t} = f(h_{f_t} + h_{b_t} + W_{f_{xh}} x_t) \tag{2.11}$$

The output units at time step, t, in Fig 2.4 and Fig 2.5

$$y_t = g(W_{hy} h_{bi_t}) \tag{2.12}$$

where; h_{bi_t} is the bi-directional hidden unit obtained from combining forward, h_{f_t} and backward h_{b_t} unit at time step, t. $W_{f_{xh}}$ and W_f are forward weight matrices; while $W_{b_{xh}}$ and W_b are backward weight matrices. W_{hy} is the output weight matrix. f and g are activation and softmax functions, respectively.

[**Ozan2014**] used deep bi-directional networks for opinion mining, where each word in the sentence has labels, while relation classification provides only a single label for the complete sentence. They stacked in depth recurrent neural network ([**Schmidhuber1992**]). [**Pascanu2013**] explore other ways of constructing deep RNNs that are orthogonal to the concept of stacking layers on top of each other. In this thesis work, we do not use stacked networks.

We investigated several variants of bi-directional recurrent networks for relation classification and also proposed novel architectures that lead to superior results for relation classification task on SemEval10 task 8 and slot filling dataset.

2.2 Relation classification

Several methods has been devised for semantic relation classification, including supervised, unsupervised and semi-supervised methods. Major differences between these methods include available resources, degree of pre-processing, features used, classification algorithm and the nature of training/test data. Supervised learning for relation classification is generally preferred over unsupervised learning due to ease of evaluation. However, annotated data is usually expensive to obtain therefore; it has been growing interest especially in the industrial domain, where annotated data is hardly available.

In *unsupervised* and *semi-supervised* paradigm, DIPRE [**Brin1998**] and Snowball [**Agichtein2000**] are the bootstrapping-based systems that used both bootstrapping framework and pattern matching framework to generate extraction patterns, which in turn result in extracting new tuples from the document collection. However, [**Zhang2004**] used bootstrapping [**Yarowsky1995**] techniques over the top of SVM for relation classification, to leverage the small amount of labeled data and large amount of unlabeled data. He used various lexical and syntactic features to extract top-level relation types in the ACE corpus. But, many key problems remained unresolved like selection of initial seed set, which is tackled by stratified sampling strategy [**Qian2009**] in the bootstrapping procedure underlying SVM as supervised classifier, where every relation type is represented in the initial seed set leading to diverse structures in the labeled set.

In 2010, manually annotated data for relation classification was released in the context of a SemEval10 shared task [**Hendrickx2010**]. Ten teams participated in the shared task, using mostly supervised classifiers such as support vector machines or maximum entropy classifiers [**Rink2012**; **Tratz2010**]. In this thesis, we also focus on supervised machine leaning techniques for relation classification. Most of the *supervised* approaches [**Miller2000**; **Zhang2006; Qian2008**; **Hendrickx2010**] for relation classification treat the task as a multi-class classification problem. Relation classification is widely studied and the existing methods mainly fall into the three categories:

1. Feature-based
2. Kernel-based
3. Neural Network-based

The focus of our research is to exploit the capabilities of neural networks, especially, convolution and recurrent neural network methods for relation classification, without

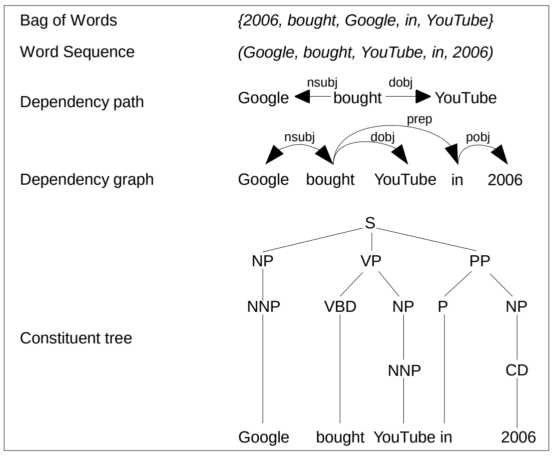

Figure 2.6: Basic relational NLP features

using NLP tools such as WordNet and without extracting expensive lexical features such as POS; and syntactic features such as chunking, syntactic parse trees, named entities, WordNet relations, etc. Maximum entropy model [**Kambhatla2004**] used to combine these features for relation classification, but the *feature based* methods are hard to generalize. *Kernel-based* methods relies on computing similarity ([**Zelenko2003**; **Mooney2005**; **Wang2008**; **Plank2013**]) between two tree by utilizing their common sub-trees and relation is mainly defined by the dependency path between target entities . Designing effective similarity measure, kernel function which captures the complete structural and semantic information, is difficult in kernel-based relation classification methods. *Neural networks* are appropriate for capturing complex interactions and compositionality.

Emerging recently, deep learning [**Bengio2009**] has captured the attention for multiple applications in natural language processing, computer vision and speech recognition. Among the different deep learning techniques, recurrent neural network and convolution neural network have been successful in various NLP tasks such as relation classification, sentiment analysis [**Kim2014**; **Santos2014**], part-of-speech tagging[**SantosandZadrozny2014**], language modeling [**Mikolov2010**], semantic role labeling [**Collobert2011**], sentence completion [**Hu2014**], etc.

Along with recurrent neural networks, *convolution neural network* has been successfully applied to relation classification task. [**Zeng2014**] proposed an approach, mentioned in Fig 3.2 (Left, Middle), for relation classification where they used word embedding (WF) and position features (PF) as input and the sentence-level features are learned

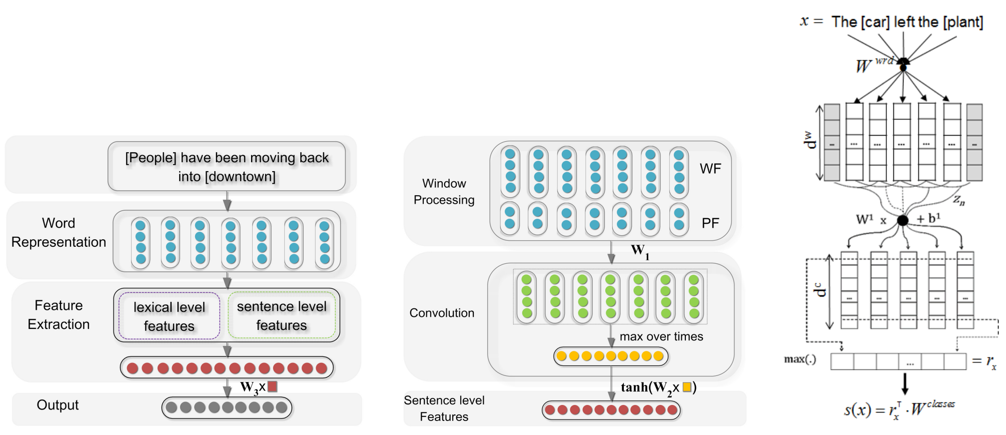

Figure 2.7: (Left, Middle): Relation Classification via CNN [**Zeng2014**], (Right): Relation Classification via CR-CNN [**Santos2015**]

through a CNN. They also extracted lexical features according to given nouns and, then sentence-level and lexical features are concatenated into a single vector and fed into a softmax classifier for relation type prediction.

The most recent work in relation classification by CR-CNN [**Santos2015**], as depicted in Fig 2.7 (Right) that achieved the state-of-art performance for the SemEval-2010 Task 8 dataset by introducing a pair-wise ranking method, while other approaches treat relation classification as multi-class classification problem by using the softmax function on the top of the CNN/RNN. CR-CNN used an effective method to deal with artificial classes by omitting their embeddings and performing classification by ranking loss on the top of CNN, while other approaches treat all classes equally. In ranking based proposed neural network, for given sentence x and two target nouns, CR-CNN computes a scores for each relation class. It learns a distributed vector representation which is encoded as a column in the class embedding matrix $W^{classes}$. As detailed in the Fig 3.2 (Right), the CR-CNN first transforms words into real valued feature vectors, then a convolution layer constructs a distributed vector representations of the sentence, r_x. Finally, CR-CNN computes a score for each relation class by performing a dot product between r_x^T and $W^{classes}$. Similar ranking approach has been introduced in the ranking-RNN in the thesis work. [**Yu2014**] proposed a Factor-based Compositional Embedding Model (FCM) by deriving sentence-level and substructure embeddings from word embeddings, utilizing dependency trees and named entities. They achieved slightly higher accuracy on the same dataset than CNN [**Zeng2014**], but only when syntactic information is used.

Recent works in relation classification and sentiment analysis via recurrent neural networks, have been proved successful and have improved the results on these NLP tasks. Recurrent neural network is a recursive deep architecture that can learn feature representation of words, phrases and sentences in the longer context. They are extensively studied in language modeling task [**Mikolov 2010**; **Mikolov 2011**] or sequence labeling [**Paliwal1997**; **Mesnil2013**]. Recently, neural networks have been applied to the shared task of relation classification on SemEval10 data set. These approaches have improved the results ([**Socher2012**; **Zeng2014**; **Yu2014**; **Nguyen2015**; **Santos2015**]) and also reduced the expensive feature extraction via NLP tools like WordNet or syntactic parse trees. These methods seem to generalize to various dataset since they does not rely much on NLP features such as POS, NER, parse trees, etc and have shown similar performance. We also investigate neural network techniques for relation classification, where we don't use any linguistics features such as POS, NER, parse trees.

The method of matrix-vector recursive neural network, MV-RNN [**Socher2012**] for relation classification used syntactic parse trees, finds the path between the two entities in parse tree, and applies compositions bottom up by recursively combining the words according to the syntactic structure of the parse tree. They then selected the highest node of the path and classify the relationship using that node's vector as features. In addition, instead of using only word vectors, an additional matrix for each word is used i.e. a matrix-vector representation to every node in a parse tree. The potential difficulty in MV-RNN approach is that they need a parser for the tree structure, hence their method is dependent on the features obtained from the NLP tools and therefore, the method is hard to generalize. We focus our research towards the generalized relation learning methods, to alleviate the need of such features derived from NLP tools.

We investigate the capability of recurrent networks to create a representation for a sequence context by processing each word in the sequence and updating the representations at each step, without any linguistic feature. However, [**ZhangandWang2015**] investigated a temporal structured RNN with only words as input. They used a bidirectional model with a pooling layer on top of it, as detailed in Fig2.8. We will also concentrate on words as input. However, we don't integrate a pooling layer directly into the model but indirectly by combining our CNNs and RNNs in the end. Furthermore, we investigate the different architectures of the RNN in more detail.

We investigate the ranking loss into RNN for relation classification and to our knowledge, this is the first attempt to introduce ranking in RNN for relation classification. We present the state-of-art performance [**Vu2016**]) combining RNNs and existing CNN for relation classification on SemEval10 Task 8 dataset.

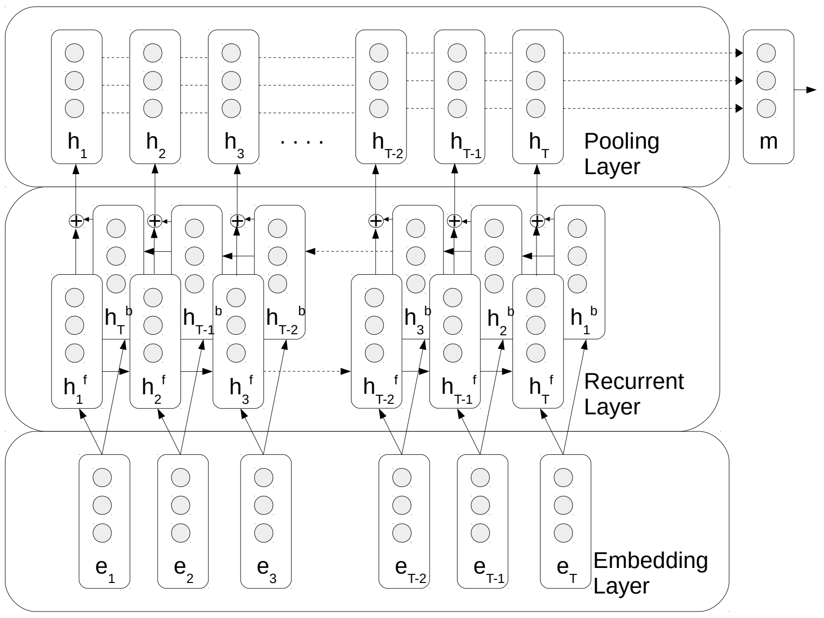

Figure 2.8: RNN with pooling for relation classification [**ZhangandWang2015**]

2.3 Slot filling

The Knowledge Base Population (KBP) track, since 2009, aims to promote research on the automated systems that discover information about names entities and incorporate this information in a knowledge source or database. Since relation classification is one of the components in the slot filling system, so in this thesis work, we focus on the English Slot Filling track, where we integrated the RNN architectures into the slot filling end-to-end system for relation classification task.

The slot filling (SF) sub-task extract the values of the specified attributes (or slots) for a given entity from the large collections of natural language texts. Examples of slot include age, birthplace and spouse for a person or founder, top members, etc. [**Mihai2014**; **Mihai2013**] gives an overview of the English slot filling (SF) track at the TAC2014 KBP evaluation. During the years, most submissions use distant supervision (DS) [**Angel2014**; **Roth2013**] method in their architecture for relation extraction. *RelationFactory* [**Roth2013**; **Roth2014**], top ranked system in TAC KBP 2013 English SF track [**Surdeanu2013**], is an open source modular relation extraction end-to-end system, from query, through retrieval of candidate contexts and judging whether a relation is expressed, to normalizing answers and putting them into a knowledge base. *RelationFactory* (Fig 2.9) is based on distantly supervised classifiers and patterns [**Roth2013**]. The highest score for 2014 submission was obtained by DeepDive [**Niu2012**; **AngeliandGupta2014**; **Angeli2014**], which is IE framework based on DS and multi-instance multi-label relation extractor, MIML-RE [**Surdeanu2012**].

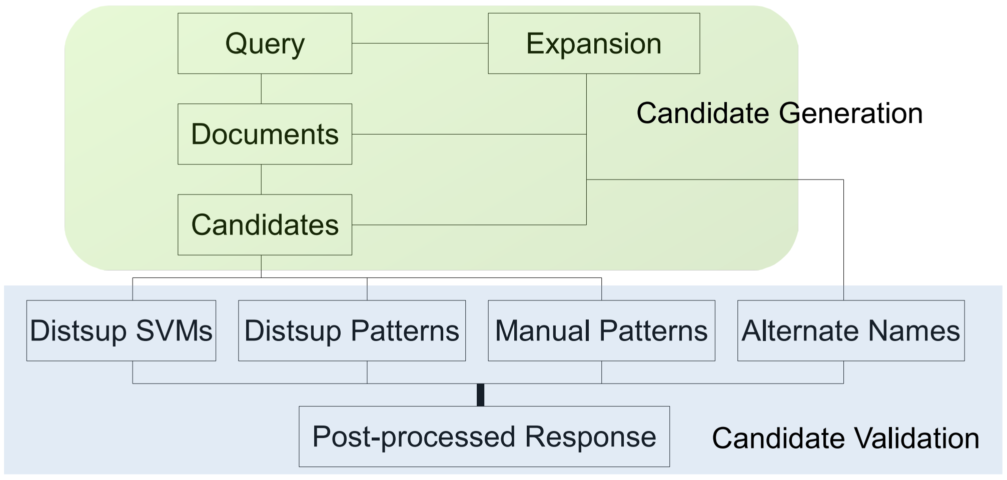

Figure 2.9: *RelationFactory* [**Roth2014**]

Most systems combined multiple approaches (e.g. DS and patterns) by simply concatenating the outputs produced by the individual components. NYU used the pattern based guidance mechanism [**Pershina2014**] to relabel relation instances during the training of the DS model. With respect to pattern-based approaches, bootstrapping methods [**Gupta2014**] were also used to acquire extraction patterns based on dependency parse paths. Distant supervision based IE systems appears to be winning, as compared to bootstrapping approaches, although DS introduces noisy examples (false positives). The TAC2014 SF evaluation [**Surdeanu2014**] suggested that the machine-learning-based approaches perform better for SF than models crafted by domain experts. Till 2014, no IE system is observed using deep learning methods for relation extraction, therefore, we investigate deep learning approach (RNNs) for slot filling and combine the results with patterns, SVMs and CNNs [**Adel2016**] for the slot filling task in TAC2015.

3 Model training and key features

3.1 Objective functions

3.1.1 Cross entropy

Most approaches use a logistic regression classifier with the softmax activation function in the final layer. The objective function which is mainly used in this case is based on cross entropy:

$$L = -\sum_{c} y_c \cdot log(s_\theta(x)_c) \tag{3.1}$$

In this equation, c iterates over all classes, y_c is the correct value for class c and $s_\theta(x)_c$ is the score the network assigned to the current data point x.

3.1.2 Ranking

In contrast to cross entropy, [**Santos2015**] trained a ranking CNN. It learns to maximize the distance between the true label y^+ and the best competitive label c_- given a data point x. The objective function is -

$$L = \log(1 + \exp(\gamma(m^+ - s_\theta(x)_{y^+}))) + \log(1 + \exp(\gamma(m^- + s_\theta(x)_{c^-}))) \tag{3.2}$$

where $s_\theta(x)_{y^+}$ and $s_\theta(x)_{c^-}$ being the scores for the classes y^+ and c^- respectively. the parameter γ controls the penalization of the prediction errors and m^+ and m^- are margins for the correct and incorrect classes. Following CR-CNN [**Santos2015**], we set $\gamma = 2, m^+ = 2.5, m^- = 0.5$. We do not learn a pattern for the class *Other* but increase its difference to the best competitive label by using only the second summand in the equation above, during training.

We introduced this ranking function, instead of $softmax$ in the RNN model and it leads to superior results on SemEval10 data set for relation classification task. For slot filling we don't use ranking, since we train an individual model, predicting positive or negative, for each slot and therefore, there are only two class predictions (0 and 1) without any artificial class. It is the first attempt to include ranking into RNNs ([**Vu2016**]).

3.2 Key features

We present the key components included in the proposed recurrent neural network models.

3.2.1 Word representation

Word embeddings are dense vectors. They can be learned based on distributional information from a large amount of text data. [**Kim2014**] and [**Grishman2015**] showed that pre-trained word embeddings outperform randomly initialized vectors and that it can be beneficial to update them during training. In this thesis work, we used pre-trained word embeddings of 50 and 400 dimensions, trained on English Wikipedia, and they are updated during model training. We used randomly initialized vectors for words not appearing in the pre-trained embedding dictionary. These randomly intitialised word embeddings are also updated during the model training. The embedding updates introduce an additional free parameter *emb* to the list of model parameters.

We observe that embedding updates perform superior to no embedding updates on the pre-trained embeddings for SemEval10 as well as slot filling data set. In our experiments with uni- and bi-directional RNNs, we used word embeddings, represented as x_t for each word w_t at each time step, t, for the input word sequence, S.

3.2.2 Target entity presence flag

In relation learning, it is essential to let the algorithm know the target nominals. Therefore, we also introduced entity presence flags, indicating whether the current word is one of the relation arguments. Two entity flags are introduced, each one for corresponding relation argument and is concatenated to word embedding representation. Therefore, the input vector at time step, t, is given by -

$$[x_t, e1\_flag, e2\_flag]$$

where, x_t is the word vector of dimension d, *e1_flag* is the flag for the first entity mention (relation argument) i.e. <e1>, while *e2_flag* is the flag for the second entity mention i.e. <e2>, in the word sequence.

For example,

S: [*The <e1>destruction</e1> is caused by the <e2>bombing</e2>.*]

where, *<e1>destruction</e1>* is the first relation argument, while *<e2>bombing</e2>* is the second relation argument. Therefore, including the entity presence flags , the input to model is represented as -

$$
\begin{gathered}
w_{the} \rightarrow [\mathrm{x}_{the}, 0, 0] \\
w_{<e1>destruction</e1>} \rightarrow [\mathrm{x}_{<e1>destruction</e1>}, 1, 0] \\
w_{is} \rightarrow [\mathrm{x}_{<e1>destruction</e1>}, 0, 0] \\
w_{caused} \rightarrow [\mathrm{x}_{caused}, 0, 0] \\
w_{by} \rightarrow [\mathrm{x}_{by}, 0, 0] \\
w_{the} \rightarrow [\mathrm{x}_{the}, 0, 0] \\
w_{<e2>bombing</e2>} \rightarrow [\mathrm{x}_{<e2>bombing</e2>}, 0, 1] \\
w_{.} \rightarrow [\mathrm{x}_{.}, 0, 0]
\end{gathered}
$$

where, w_t is the word at time step, t, while x_t is the corresponding embedding vector for word, w_t.

3.2.3 Word position features

In the task of relation classification, information that is needed to determine the class of a relation between two target nouns normally comes from words which are close to the target nouns. [**Zeng2014**] propose the use of word position embeddings (position features) which help the CNN by keeping track of how close words are to the target nouns. These features are similar to the position features proposed by [**Collobert2011**] for the Semantic Role Labeling task.

In the thesis work, we incorporate randomly initialized word position embeddings, position features (PF), similar to [**Zeng2014**], [**Grishman2015**] and [**Santos2015**]. These are special embeddings which indicate the relative position (distance) of the current word to the two relation arguments. We also consider if the word is to left or right of the relation arguments by '+' and '-' sign with the distance. For instance, in example sentence S above, the relative distances of *is* to *<e1>destruction</e1>* and *<e2>bombing</e2>* is 1 and -4.

Each relative distance is mapped to a vector ([d_e1] and [d_e1]) of dimension, d_e (a hyperparameter), which is randomly initialized. The position embedding of w_t is given by the concatenation of these two vectors, $PF = [d\_e1, d\_e2]$. with random numbers. Since, the position embeddings are updated by index of the word in the dictionary therefore, during our implementation , we shifted the position index of each word by the minimum distance of a word to a relation argument in a sentence for the whole corpus so that the negative position values could also be represented in the position embedding vector. We used these position features (PF) along with entity presence flag for each input word, as pictured later in Fig 4.2. Therefore,

$$w_t \rightarrow [\mathrm{x}_t, \textit{e1\_flag}, \textit{e2\_flag}, \textit{PF}]$$

Note that, the concatenation of position features and entity presence flags to the word vectors alters the size of input.

3.2.4 Position indicators

Alternatively to position features (PF), we also investigated *position indicators* (PI) in recurrent neural networks as proposed by [**Zhang2015**]. In the CNN-based approach, [**Zeng2014**] appended a position feature vector to each word vector, i.e., the distance from the word to the two nominals. This has been found highly important to gain high classification accuracy. For RNN, since the model learns the entire word sequence, the relative positional information for each word can be obtained automatically in the forward or backward recursive propagation. It is therefore sufficient to annotate the target nominals in the word sequence, without necessity to change the input vectors, while increases the length of the input word sequences, as four independent words, as position indicators ('<e1>', '</ e1>', '<e2>', '</e2>') around the relation arguments. For instance, the input word sequence S in represented below, including the position indicators, (as in Fig. 4.3):

$S_{PI} \rightarrow$ {*the* <e1> **destruction** </e1> *is caused by the* <e2> **bombing** </e2>. }

The four position indicators which are regarded as single words are introduced in the training and testing sets around the two nominals e.g. **destruction** and **bombing** in the sentence, S. The position embedded sentences are used as the input to train the RNN model. Compared to the position feature (PF) approach in the CNN model, the position indicator method is more straightforward and performs superior to position features. Also, it does not introduce additional free parameter, i.e. position embedding dimension and does not alter the size of input vector to the RNN model.

PI are interpreted as :

- '<e1>' : a word before the first relation argument in the word sequence
- '</e2>' : a word after the first relation argument in the word sequence
- '<e2>' : a word before the second relation argument in the word sequence
- '</e2>' : a word after the second relation argument in the word sequence

3.2.5 Extended context

During the model training and testing process, we also take the extended context around the target relation arguments, instead of only the context in-between the nominals in the sentence. The contexts used in training and test is accumulated from three regions: the left context of first relation argument, the middle context in-between the two relation arguments and the right context to the second relation argument. For SemEval10 data set, we take 5 neighboring words to the left of first nominal, the context in-between the target nominal including the two nominals and 5 words neighboring to the right of second nominal. Since, slot filling data set can have multiple relation arguments (nominals) in the same sentence therefore, we take the 5 left and right neighbor words to the extreme occurrences of nominals in the sentence.

Our investigation found that the middle context expresses the main information of the relation in most cases, discussed in the semantic meaning accumulation by RNNs.

3.2.6 Word features

Distributional hypothesis theory ([**Harris1954**]) indicates that words that occur in the same context tend to have similar meanings. To capture this characteristic, the word features (WF) combines a word's vector representation. Initial experiments showed that using trigrams as input instead of single words leads to superior results. Hence at timestep t, we do not only give word, w_t to the model but for instance the trigram $w_{t-1}w_tw_{t+1}$ by concatenating the corresponding word embeddings. We demonstrate the 3-gram input to RNN in Fig. 4.4.

In our experiments with word features (WF) for SemEval10 dataset, we observed that 5-gram input to the RNN model leads to superior results.

3.3 Optimization

Training a recurrent neural network is tricky and a number of factors are considered during optimization, as given discussed below: Training the BRNN model in Fig 4.10 involves optimizing the parameters

$$\theta = [W_f, U_f, W_b, U_b, W_{bi}, W_{hy}, emb] \tag{3.3}$$

where emb is embedding matrix, $emb \epsilon R^{N \times d}$. N is the total number of words and d is the embedding dimension of each dense word vector. In case of position features and word features, the two additional parameter, position embedding dimension (d_e) and context window size (w) are also added to the parameter list θ.

The training objective is that, for a given sentence, the last output feature vector h_{bi} in Fig 4.10, achieves the best performance on the task of relation classification. Here we use a simple logistic regression model using softmax and ranking, as discussed above.

To train such a model, we utilizes the stochastic gradient descent (SGD) algorithm, where the gradient is computed at sentence level. Specifically, the back propagation

```
Pseudo-code for norm clipping
$\hat{\mathbf{g}} \leftarrow \frac{\delta\varepsilon}{\delta\theta}$
if $||\ \hat{\mathbf{g}}\ || \geq$ threshold then
        $\hat{\mathbf{g}} \leftarrow \frac{threshold}{||\mathbf{g}||}\hat{\mathbf{g}}$
end if
```

Table 3.1: Pseudo-code for norm clipping

through time (BPTT) [**Werbos1990**] is employed to compute the gradients layer by layer and error is backpropagated, once for each sentence, through time from the end of the sentence where the relation lable is attached. However, training RNNs by using back-propagation through time to compute error-derivatives can be difficult. Early attempts suffered from vanishing and exploding gradients [**Bengio1994**] and this meant that they had great difficulty learning long-term dependencies. Many different methods have been proposed for overcoming this difficulty. Here, we briefly discuss our model training and few tricks that we used for optimization of recurrent networks for capturing the long term dependencies.

3.3.1 Exploding and vanishing gradients

The goal of a RNN implementation is to enable propagating context information through faraway time-steps. Recurrent neural networks propagate weight matrices from one timestep to the next. During the back-propagation phase, the contribution of gradient values gradually vanishes as they propagate to earlier time-steps. Thus, for long sentences, the probability that relation-type would be recognized reduces with the larger size of context lengths.

Introduced in [**Bengio1994**], the exploding gradients problem refers to the large increase in the norm of the gradient during training. Such events are due to the explosion of the long term components, which can grow exponentially more than short term ones. The vanishing gradients problem refers to the opposite behavior, when long term components go exponentially fast to norm 0, making it impossible for the model to learn correlation between temporally distant events. The mechanics is discussed in detail by [**Pascanu2013**].

To deal with exploding gradient, we use an L2 penalty on the recurrent weights, where L2 weight set is 0.0001. *Gradient Norm Clippoff* [**Mikolov2010**], the mechanism to deal with the exploding gradient problem is to rescale their norm whenever it goes over a threshold. The Pseudo-code for norm clipping is given in Table 3.1.

The *Gradient Norm Clippoff* is simple and computationally efficient, but it does however introduce an additional hyper-parameter, namely the threshold. We found the threshold of 10.0 from validation during training. To deal with *vanishing gradient problem*, we use *Relu and CappedRelu* activation units, as discussed below.

3.3.2 Activation functions

Learning long term dependencies in recurrent networks is difficult due to vanishing and exploding gradients. To overcome this difficulty, researchers have developed sophisticated optimization techniques and network architectures. In our experiments, we apply three different types of nonlinearities to the hidden units, as -

1. **Sigmoid function** : Sigmoid function, Fig 3.1 (Left), refers to the special case of

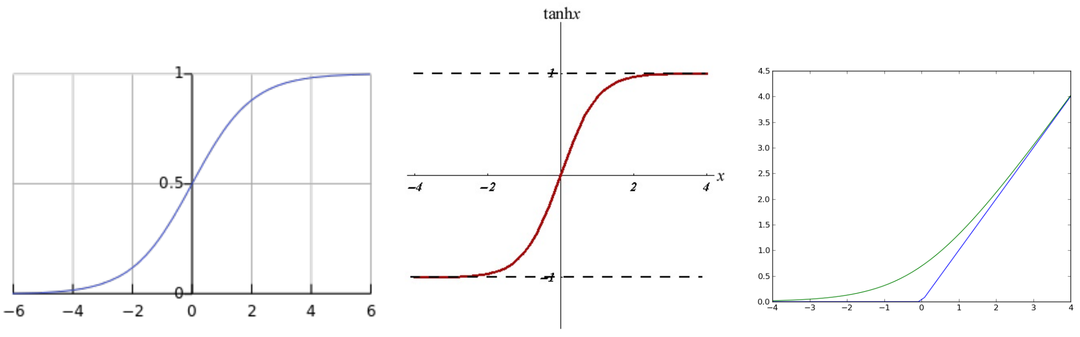

Figure 3.1: Sigmoid (Left), Tanh (middle), and ReLu (right) activation functions

   the logistic function, having an "S" shape.

$$S(x) = \frac{1}{(1 + \exp(-x))} \tag{3.4}$$

   where, $S(x)$ is the output of sigmoid function for the given input. The disadvantage of sigmoid activations is that fitness landscape would look less smooth, therefore, a small change in the weights may result in a big change in the output i.e. explode gradient or no change at all i.e. vanishing gradient problem. The sigmoid activation function has the potential problem that it saturates at zero. So if the activity in the network during training is close to zero then the gradient for the sigmoid activation function may go to zero.

2. **Tanh function** : As compared to *sigmoid*, *tanh*, Fig 3.1 (Middle), saturates at plus and minus one. It turns out that the logistic sigmoid can cause a neural network to get “stuck” during training. This is due in part to the fact that if a strongly-negative input is provided to the logistic sigmoid, it outputs values very near zero. Since neural networks use the feed-forward activations to calculate parameter

gradients, this can result in model parameters that are updated less regularly than we would like, and are thus "stuck" in their current state. Therefore, *tanh* has advantage that the strongly negative inputs will map to negative outputs. Additionally, only zero-valued inputs are mapped to near-zero outputs. These properties make the network less likely to get "stuck" during training.

$$S(x) = \frac{1}{(1 + \exp(-x))} \tag{3.5}$$

In our experiment, we also used *tanh* activation function for slot filling and SemEval10 data sets for relation classification.

$$g_{tanh}(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{3.6}$$

3. **Rectified Linear Units (ReLu) function** : Given the input x, the output of Relu unit is a ramp function (Fig 3.1 [Right]) , as -

$$y = max(x, 0) \tag{3.7}$$

where, y is the output of rectified linear unit.

4. **Capped Rectified Linear Units (CappedReLu)function** : CappedReLu is an extension of Relu, where the units are capped by a threshold. It was also investigated in Deep Belief Networks [**Alex2010**] for image classification task. Therefore, mathematically,

$$y = min(max(x, 0), t) \tag{3.8}$$

where t is the capping threshold. We set $t = 6$ in all our experiments. The CappedRelu encourages the model to learn sparse features earlier.

ReLu and CappedRelu observed to stabilize the RNN training, since they offer sparse activation (For example, in a randomly initialized networks, only about 50% of hidden units are activated, having a non-zero output) and also reducing the vanishing gradient problem or exploding effects. Rectified linear units, compared to sigmoid function or tanh activation functions, allow for faster and effective training of neural networks. As discussed by [**Glorot**], a great care must be taken when initializing the weights for the network so that the gradient doesn't go to zero during the first epoch of training.

3.3.3 Model initialization strategies

We observe that, with the right initialization of the weights, RNNs composed of rectified linear units are relatively easy to train and are good at modeling long-range dependencies. As proposed by [**Quoc2015**], we initialize the recurrent weight matrix to be the identity matrix and biases to be zero. This means that each new hidden state vector is obtained by simply copying the previous hidden vector then adding on the effect of the current inputs and replacing all negative states by zero. The identity initialization has the very desirable property that when the error derivatives for the hidden units are backpropagated through time they remain constant provided no extra error-derivatives are added.

The combination of *identity weight matrix initialization* with *rectified linear units and cappedRelu* leads to superior results as compared to weights initialized by uniform or Gaussian distribution. The sparsity in weight matrix helps improving the optimization, specially in recurrent neural networks.

3.3.4 Learning rate decay strategies

During training via SGD, we implemented several learning rate decay schedules, as mentioned below:

- Decay learning rate by *decay_rate* at each epoch, update: $lr = lr * decay\_rate$
- Decay learning rate by *decay_rate* as defined after *n_epochs* > 1, update: $lr = lr * decay\_rate$
- Decay learning rate at each epoch, update: $lr = lr/t$, where t is the current epoch number,
- Decay learning rate by *decay_rate* as defined after 'n_epochs' > 1, update: $lr = lr * decay\_rate$, based on the development set scores. Check for development set score for *n_epochs*. Update the learning rate if the score does not improve after *n_epochs* and reload the saved model. A model is saved when the development set score is the best obtained so far from all previous epochs.

Learning Rate Decay Strategy 4 with *decay_rate* = 0.5 after *n_epochs* = 3 makes the RNN learning more stable.

4 Recurrent neural networks for relation classification

We discussed in detail the conventional RNN models for sequence modeling in Chapter 2, and here we investigate the capability of Uni-and-bi-directional RNNs in capturing the semantic meaning of word sequence for relation classification task, where the significant difference of our relation classification models is that there are no predicting targets at the each time step, and the supervision (relation label) is only available at the end of the sequence, i.e. sentence-level relation classification. We have proposed several modification in the architectures that enable us to obtain the state-of-art performance on relation classification task via RNNs. The thesis contribution includes the implementations of all the following RNN variants in Theano, and their training and evaluation for slot filling (SF) as well as SemEval10 task8 data set for relation classification.

4.1 Uni-directional RNNs

In contract to RNN architectures for sequence classification, we have output vector (relation label) at the end of the sentence. Therefore, the semantic meaning of the word sequence is accumulated by the recurrent units h_{f_t} at each time step word-by-word and this history is propagated to the end where supervision is performed. The history h_{f_t} at time-step t is accumulated by conditioning on all the information of the preceding words, which is accumulated in h_{f_t-1}. We investigate the capability of Uni-directional RNNs for relation classification, as mentioned in Fig 4.1, where the history vector h_{f_N} captures the *semantic meaning* of the input example and is responsible for predicting relation label. Let's take an example sentence for relation *cause-effect(e2,e1)*:
S : *The <e1>destruction</e1> caused by the <e2>bombing</e2>*
The mathematical representation is similar to Eq.(4.1) and Eq.(4.2), except the output is predicted once at the end of the sentence, and is given by :

$$h_{f_N} = f(W_{xh}x_N + W_{hh}h_{f_{N-1}}) \tag{4.1}$$

$$RELATION = g(W_{hy}h_{f_N}) \tag{4.2}$$

where;

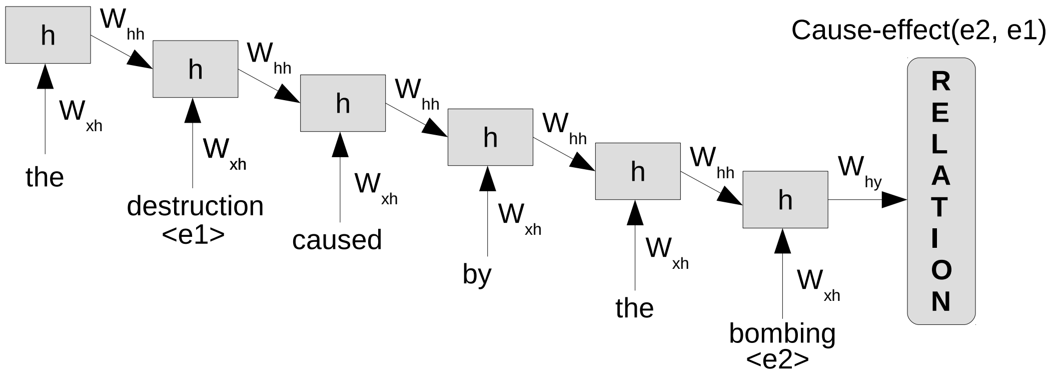

Figure 4.1: Uni-directional RNN for relation classification.

1. $\{x_1, x_2, x_3 ... x_N\}$: the input word sequence {*the, destruction, caused, by the bombing*}
2. x_N : the last input word vector of word sequence; $x_N \epsilon$ 50 for SF task and $x_N \epsilon$ 50, 100 for SemEval10 task8
3. h_{f_N} : the history at the end, accumulating semantic meaning of complete sentence
4. $RELATION$: the output relation predicted at the end of word sequence

These recurrent networks in Fig 4.1 heavily suffer from vanishing and exploding gradient [**Bengio1993**] difficulties, since the network computes the output vector once for each sentence and the error is backpropagated from the relation-label way back to the first word in the input word sequence. Therefore, optimization of such network is more tricky as compared to tradition RNN which computes the output vector and error is backpropagated at each time-step for the input word. To avoid vanishing gradient problem, we use gradient norm clipping [**Pascanu2012**] for all the RNN architectures for relation classification.

The input to the recurrent neural network in Fig 4.1 is sequence of word vectors, processed word-by-word in the unfolded recurrent computations. The *word vectors* are represented by *1-hot encoding* or *word embeddings* [**Kim2014**; **Mikolov2013**], initialized randomly or pre-trained. In this thesis work, the words are represented by a pre-trained word embeddings of size 50 for slot filling and {50,400} for SemEval10 task8.

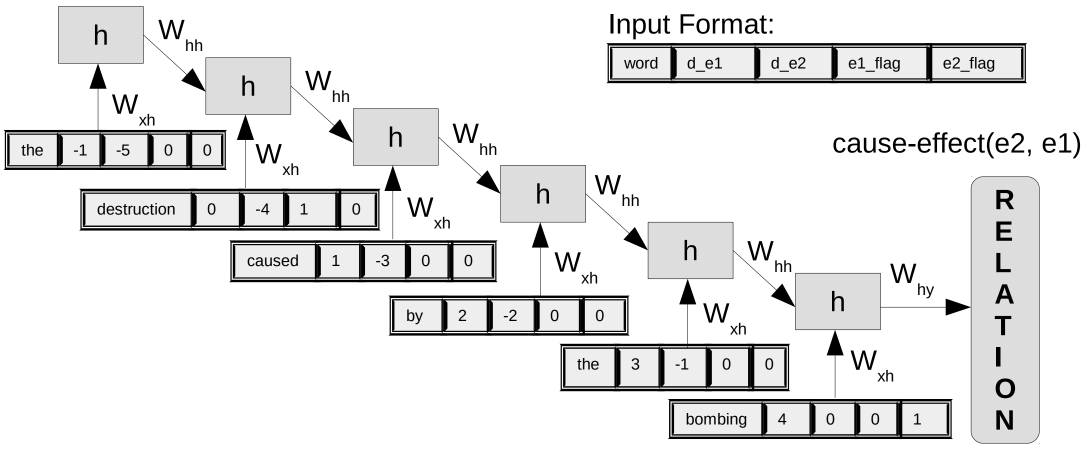

Figure 4.2: Uni-directional RNN for relation classification with position embeddings and entity mention flag.

4.1.1 Uni-directional RNN with position features

We performed initial experiments with only a single word (word vector), but we do experiment with various position features for each word, as input along with word vectors which improved the classification performance. Since relation classification is complex task and it is not possible to capture the structured features (e.g. shortest dependency path between nominals) only through word features (WF), as explained in section 3.2.6.
Therefore, we use different possibilities of position features (PF) for RNN, as :

1. position embeddings, refer section 3.2.3
2. position embeddings with entity presence flag, in (Fig. 4.2)
3. position indicators, refer section 3.2.4

Fig. 4.2 describes the PF, where we take position embedding and entity mention flags. This flag indicates if the current word is one of the relation argument i.e. relation mention, in the sentence, S. For the *input format*, as described in the Fig. 4.2, we have:

1. *word* : the input word vector (word embedding, section 3.2.6) in the current word sequence; $x_N \epsilon$ 50 for SF task and $x_N \epsilon$ 50, 100 for SemEval10 task8
2. d_e1 : relative distance (position embedding, section 3.2.3) of the current *word* to *<e1>* i.e. *destruction*

3. d_e2 : relative distance (position embedding, section 3.2.3) of the current *word* to <*e2*> i.e. *bombing*
4. e1_flag : entity mention flag of the current for *word* if it is <*e1*> type argument
5. e2_flag : entity mention flag of the current for *word* if it is <*e2*> type argument

The relative distance (*in terms of number of words*) from <*e1*> (first relation argument) and <*e2*> (second relation argument) for each word in the sentence is computed and it is also consider if the current word is preceding or following, to relation arguments, by *sign* (+/-). Since, in slot filling case, we have multiple slots and fillers (i.e. relation arguments) so we take the relative distance to the closest relation argument (or nominal) for each word in the sentence. For example, the relative distance of *caused* in the sentence S to *destruction* and *bombing* is 1 and -3 respectively. If the current word e.g. *destruction* is one of the relation argument then it's entity flag e1_flag is marked 1 corresponding to the relation argument type i.e. of *e1* type, while e2_flag is marked 0. The relative distance of *destruction* from <*e1*> and <*e2*> is 0 and -4 respectively. In our method, these relative distances are also mapped to a vector of dimension d_e (a hyperparameter) and this vector is randomly initialized as described in section 3.2.3 . Therefore, mathematically, the input x_t is represented by -

- PF : [d_e1, d_e2]
- WF : [word embedding (dimension: d) for word at time, t]
- x_t : (WF, PF, e1_flag + e2_flag); $x_t \epsilon \ (d + d_e + d_e + 2)$

Combining the WF, PF and entity mention flags, the input (Fig. 4.2) is subsequently fed into the recurrent neural network, instead of only the word embedding (WF), lead to superior results.

4.1.2 Uni-directional RNN with position embeddings

Another possibility for position features (PF) as discussed in section 3.2.4, which in contrast to position embeddings along with entity mention flag, does not change the input vector. As it is evident from Fig 4.2 and Fig 4.3, the input vector size is changed in former, while it remains unchanged in the later. Since, RNN learns the entire word sequence, the relative positional information for each word can be obtained automatically by forward or backward propagation. Therefore, a simple approach proposed by **[Zhang2015]** to annotate the relation arguments in the word sequence, without necessity to change the input vector. In our implementation, we introduced 4

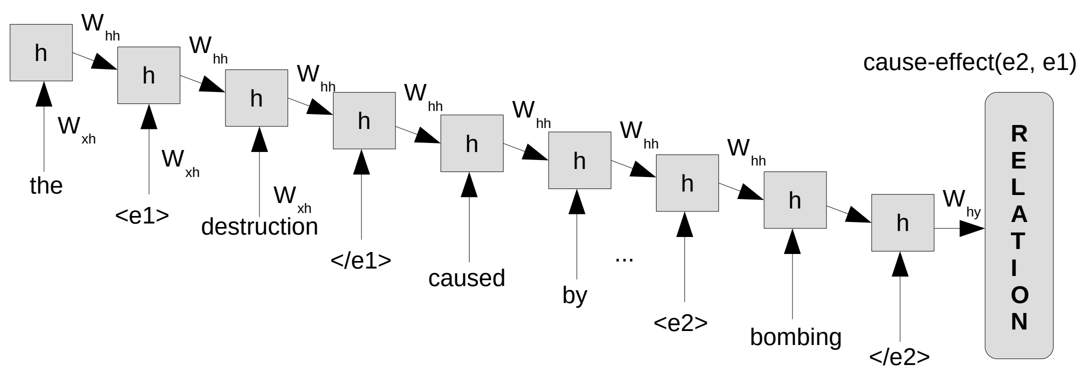

Figure 4.3: Uni-directional RNN for relation classification with position indicators.

position indicators ('<e1>', '</e1>', '<e2>', '</e2>') in a sentence, regarded as single independent words. Fig 4.3 describes the introduction of position indicators in sentence, S, without changing input vector size, which remain same as the word embedding size. For these position indicators, embeddings are randomly initialized and updated during training. The explanation of PI is given in section 3.2.4.

Though, position indicators increases the context length by introducing 4 additional words, but they help in reducing the number of free parameters i.e. position embedding size d_e for each relation argument type as well as dimensions of W_{xh} remains unchanged, leading to faster training. We have also observed that position indicators embeddings approach performs better than position embeddings, and are trained more easily (because they occur more often than embeddings of relative positions) for relation classification tasks on SF and SemEval10 data sets.

4.1.3 Uni-directional RNN with word features

Following the investigations on single word input along with position features and entity flags, we also performed experiments with n-gram (3 or 5) word vectors, instead of a single word vector as input to RNN. This word-context window, as n-gram input, helps to capture short-term temporal dependencies and patterns. With each word mapped to an embedding vector, the word-context window is the ordered concatenation of word embedding vectors, as described in Fig 4.4 and discussed in section 3.2.6. These word features (WF) follows the Distributional hypothesis theory [**Harris1954**], which states that words that occur in the same context tend to have similar meaning. These WF are also investigated by [**Zeng2014**], as input to convolution neural network. In this thesis work, we proposed to use these WF as input to RNN for relation learning.

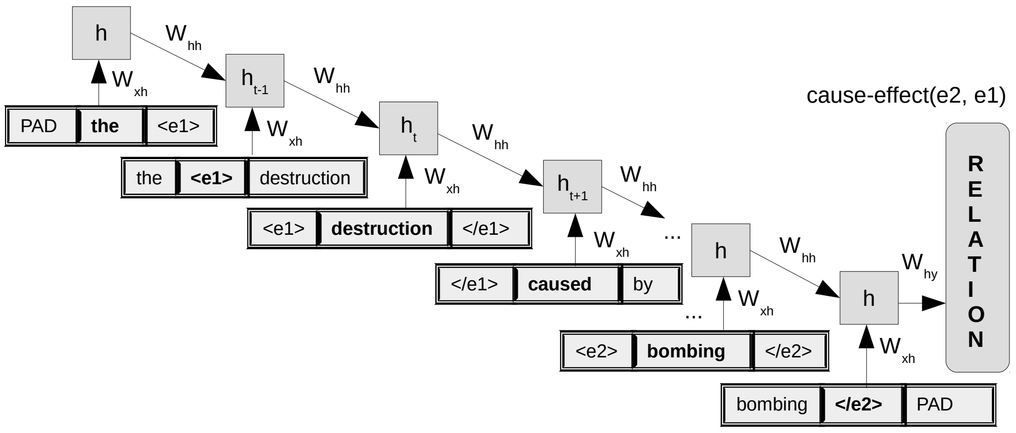

Figure 4.4: Uni-directional RNN for relation classification with Word features.

Here, in the example, the word context window of size (w) 3 is constructed:

$$\text{i.e. } w_t = [\, < /e1 >, caused, by]$$
$$'caused' \rightarrow x_{caused} \,\epsilon\, R^d$$
$$x_t = \left[x_{</e1>}, x_{caused}, x_{by}\right] \,\epsilon\, R^{3d}$$

In the example Fig 4.4, w_t is the 3-word context window around the *t-th* word *'caused'*, x_{caused} is the embedding vector of the word *'caused'*, and d is the dimension of the embedding vector. Correspondingly, x_t is the ordered concatenated word embeddings vector for the words in w_t. Here, we included position indicators (section 3.2.4) as independent word vectors for each relation argument in the word sequence.

Since, the size if input vector increases which leads to slower training, however, the n-gram (tri-gram and 5-gram) input word vectors leads to superior results than a single input word vector.

4.2 Bi-directional recurrent neural networks (BRNNs)

The bi-directional RNNs are the key part of this thesis for capturing long-distance patterns. First, we start investigating the Uni-directional recurrent neural networks, which have a potential drawback that it does not utilize the full information given. The history (Eq. 4.4) at any time step t or in middle of the word sequence at time-step, $t = \frac{N}{2}$, represents the context accumulated from the past words therefore, it can be regarded as local segment level history produced by the word segment $(x_1, x_2, ..., x_t)$. The information of the future words is not utilized and not accumulated in the history at time-step, t, when predicting the semantic meaning in the middle of a sentence. Bi-directional RNNs handles the difficulty of Uni-directional RNNs, that makes predictions using both future and past words. The bi-directional RNN for sequence learning [**Paliwal1997**], demonstrated the superiority of bi-directional RNN over Uni-directional architectures.

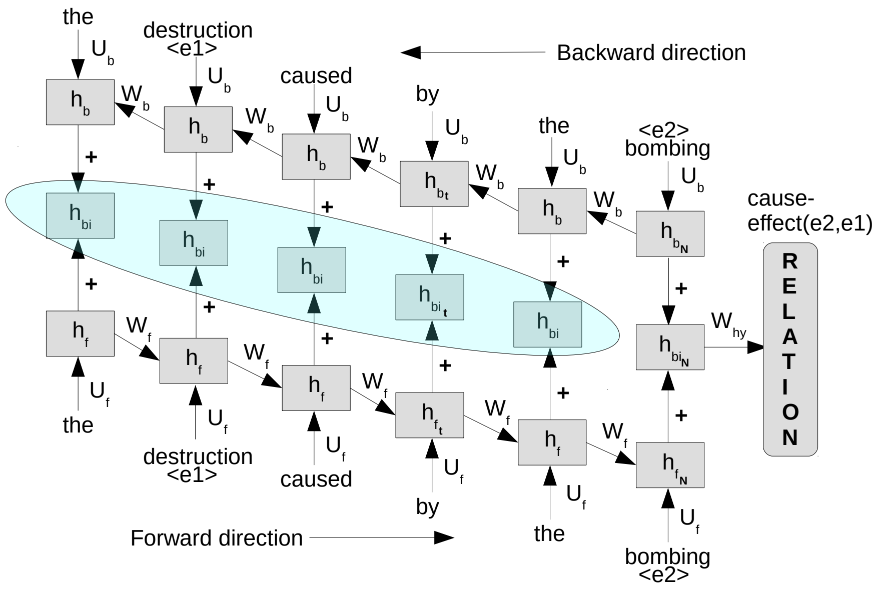

Figure 4.5: Bi-directional RNN for relation classification (no weights for combining forward and backward hidden representation, symmetric input). The highlight shows that the hidden units encapsulated are irrelevant.

Similar to Uni-directional RNNs, we have different architecture of bi-directional RNNs for relation classification task as compared to sequence learning, where we have labels at the end of the word sequence and supervision is done once for each sentence. Here, we investigate and propose various bi-directional architectures to fully exploit the information at every history node for each word input from the sequence. In the bi-directional RNN architectures, the forward and backward pass trained simultaneously by combining the forward and backward presentations.

We will investigate two architectures in regards to relation classification, where we have relation label at the end of the sentence and gradient is sentence-level. As discussed in section 2.1.2, observe that in Fig 2.4 and Fig 2.5, the forward and backward representation are combined without using weights since we found this approach is superior than using weights, in relation classification. Therefore, here, we described the networks without weights to combine forward and backward histories at each time point; while, [**Mesnil2013**] used weights to combine them. Also, we investigate other strategies to use the shared weights in forward and backward pass. Here, in Fig 2.4 and Fig 2.5, we mentioned the standard approach, where the weights in forward and backward ($W_f \neq W_b$) network are independent of each other. We shall also investigate networks where the weights in forward and backward networks are shared ($W_f = W_b$) or shared-transpose (W_f in forward and W_f^T in backward).

4.2.1 Need of BRNNs

We demonstrate the *need of bi-directional RNNs (BRNN)* by the following example, for word sequences, from SemEval10 dataset :

S_3: *"... <e1> caused <e2> ..."*, R_3 = *cause-effect(e1, e2)*
S_4: *"... <e1> caused by <e2> ..."*, R_4 *cause-effect(e2, e1)*

BRNN accumulates the semantic meaning by composition of forward and backward pass at the current word input e.g. *caused*. Thus, for relation type R_3 for word sequence S_3, at input word vector *'caused'*, the backward pass offers future context that is conditioned on the immediate word '<e2>' ; while, for relation type R_4 for word sequence S_4, at the same input word vector *'caused'*, the backward pass offers future context that is conditioned on the immediate word *'by'*. Therefore, the composition of past and future accumulated context at input word *'caused'*, captures the directionality in the relation types and, hence help in effective pattern and relation learning via BRNNs.

4.2.2 BRNNs step-by-step

In the chapter 2, we discussed in detail the bi-directional RNNs for sequence classification, where we have labels for each word in the sentence. Here, we discuss our investigations on bi-directional RNNs for relation classification task. We propose several architectures which leads to superior results than the standard RNN architectures. Our investigations are based on the bi-directionality, weight sharing, combination of forward and backward networks, and long-term dependencies capturing by BRNNs. We also proposed the *Connectionist* bi-directional network and introduced ranking RNNs. These investigations for relation classification task on SemEval dataset as well as in slot filling, contribute to the novelty components that we bring to this thesis work.

For forward network, as discussed in Uni-directional RNNs, the current word is conditional on the past words. While, for backward network, the current word is conditioned on the future words in the word sequence. The steps for bi-directional network for relation classification, are as explained below:

1. ***F*ORWARD PASS**: The forward hidden units (h_f) are computed word-by-word at each time step. The current word is conditioned on the past words in the word sequence. The computed forward hidden units are stored. The forward hidden unit, as in Fig 4.5 and Fig 4.6 at time step, t, is given by -

$$h_{f_t} = f(U_f x_t + W_f h_{f_{t-1}}) \tag{4.3}$$

   where;

   - $x_0, x_1, x_2, ..., x_N$: the input word sequence, i.e. *word vectors of dimension, d,* of length, N.
   - x_t : the input word at time step, t, from the word sequence; $x_t \epsilon R^d$.
   - U_f : the weights matrix between hidden units and input in forward network, used to condition the input word vector, x_t; $U_f \epsilon R^{d \times D_h}$ and D_h is the hidden unit dimension.
   - W_f : the weights matrix connecting hidden units in forward network, used to condition the output of the previous time-step, $t-1$; $W_f \epsilon R^{D_h \times D_h}$.
   - $h_{f_{t-1}}$: the forward hidden unit at time step $t-1$, conditioned on the past words $(t_0, t_1, ..., t_{t-2})$ in the word sequence of length, N.
   - h_{f_t}: the forward hidden unit computed at time step, t, accumulating the semantic meaning given the input word x_t and history, $h_{f_{t-1}}$; $h_{f_t} \epsilon R^{D_h}$.
   - f : the activation or non-linearity function at the hidden layer, h_f. The choices are *sigmoid, tanh, rectified linear units (ReLu), cappedReLu*.

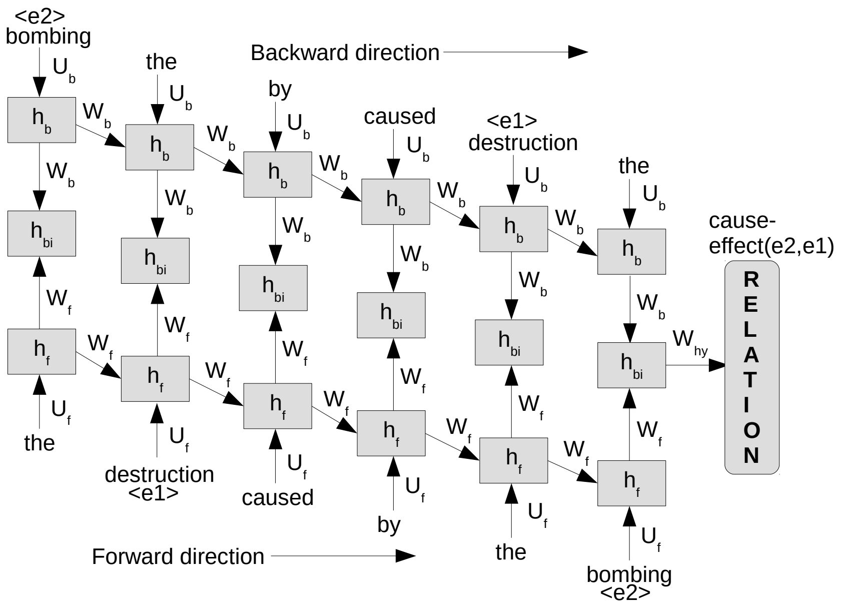

Figure 4.6: Bi-directional RNN for relation classification (weights for combining forward and backward hidden representation).

2. ***B*ACKWARD PASS**: To compute the backward hidden units (h_b), the word sequence is reversed and forward propagation, as in step 1, is made on the reversed word sequence. Here, one could observe that the current word is conditioned on all the future words due to word sequence reversal. These backward units computed are also stored. Observe that no updates/supervision are made so far, since relation label is not attached to these forward or backward units.

   The backward hidden unit, as in Fig 4.5 and Fig 4.6, at time step, t, is given by -

$$h_{b_t} = f(U_b x_t + W_b h_{b_{t+1}}) \tag{4.4}$$

   where;

   - U_b : the weights matrix between hidden units and input in backward network, used to condition the input word vector, x_t; $U_b \epsilon R^{d \times D_h}$ and D_h is the backward hidden unit dimension.
   - W_b : the weights matrix connecting hidden units in backward network, used to condition the output of the future time-step, $t+1$; $W_b \epsilon R^{D_h \times D_h}$.

- $h_{b_{t+1}}$: the backward hidden unit at time step $t+1$, conditioned on the future words (x_{t+2}, x_{t+3}, ..., x_N) in the word sequence of length, N.
- h_{b_t} : the backward hidden unit computed at time step, t, accumulating the semantic meaning given the input word x_t and the future context, $h_{b_{t+1}}$, that is accumulated and conditioned on the future words (x_{t+1}, x_{t+2}, x_{t+3}, ..., x_N); $h_{b_t} \epsilon R^{D_h}$.

3. ***C*OMBINE FORWARD-BACKWARD NETWORKS, PERFORM SUPERVISION**: The forward (h_{f_t}) and backward (h_{b_t}) units at each time step t through the word sequence length, is concatenated to obtain h_{bi_t}. The relation label is attached to the last combined hidden unit, (h_{bi_N}, N=words in input sentence), therefore, classification is performed at h_{bi_N} and error is backpropagated in the network (Fig 4.9). The forward and backward network combination, as in Fig 4.6 with weights used for combining forward and backward network, is computed as -

$$h_{bi_t} = f(W_f h_{f_t} + W_b h_{b_t}) \tag{4.5}$$

$$h_{bi_N} = f(W_f h_{f_N} + W_b h_{b_N}) \tag{4.6}$$

while, the forward and backward network combination, as in Fig 4.5 and Fig 4.7, where there are no weighted combinations, is computed as -

$$h_{bi_t} = h_{f_t} + h_{b_t} \tag{4.7}$$

$$h_{bi_N} = h_{f_N} + h_{b_N} \tag{4.8}$$

$$RELATION = g(W_{hy} h_{bi_N}) \tag{4.9}$$

where;

- W_f : the weights matrix connecting hidden units in forward network as well as h_f and h_{bi}; $W_f \epsilon R^{D_h \times D_h}$. For our experiments, hidden unit dimensions, D_h, for forward, backward and concatenated hidden units are same.
- W_b : the weights matrix connecting hidden units in backward network as well as h_b and h_{bi}; $W_b \epsilon R^{D_h \times D_h}$.
- h_{f_t} : the forward hidden unit computed at time step, t, accumulating the semantic meaning given the input word x_t and history, $h_{f_{t-1}}$; $h_{f_t} \epsilon R^{D_h}$.
- h_{b_t} : the backward hidden unit computed at time step, t, accumulating the semantic meaning given the input word x_t and the future context, $h_{b_{t+1}}$, that is accumulated and conditioned on the future words (x_{t+1}, x_{t+2}, x_{t+3}, ..., x_N); $h_{b_t} \epsilon R^{D_h}$.

- N : Number of words in the sentence
- h_{bi_N} : the concatenated last hidden unit, where supervision is performed, and is obtained from combination of last hidden units from forward and backward networks; $h_{bi_N} \epsilon R^{D_h}$.
- h_{f_N} : the forward history at the end, accumulating semantic meaning of complete sentence by conditioning each word on the past words in the word sequence.
- h_{b_N} : the backward history at the end, accumulating semantic meaning of complete sentence by conditioning each word on the future words in the word sequence.
- W_{hy} : the output weight matrix ; $W_{hy} \epsilon R^{D_h \times C}$.
- g : the softmax function at the output layer; where $g(z_m) = \frac{\exp(z_m)}{\sum_k \exp(z_k)}$
- $RELATION$: the output relation predicted at the end of word sequence

We also investigated the combination forward and backward network *without* weights between h_f and h_{bi} as well as h_b and h_{bi}, as in Fig 4.7, which leads to superior results as compared to weighted combination in Fig 4.6. Along with the difference in (weights) combination in Fig 4.5 and 4.6, one can observe the difference in the order of hidden units from backward network (also in Fig 4.5 and Fig 4.7) are used in concatenation to hidden units of forward network. The two ways of concatenating the forward and backward networks using different orders of hidden units, is discussed below in this section.

4.2.3 Forward/backward network combinations

Here, we briefly discuss the two variants of bi-directional RNNs, based on the order of backward hidden units used to concatenate with forward units.

1. As in Fig 4.7, the orange colored line indicates the context flow from forward and backward network to the concatenation to obtain h_{bi}. The forward hidden unit attached to the first word is concatenated to the backward hidden units attached to the last word in the sequence. Therefore, no full information is utilized at h_{bi_0}. Subsequently, the second concatenated hidden unit, h_{bi_1} has context representation of -
    - (*destruction* | *the*) i.e. *destruction* conditioned on *the*, from forward network,
    - (*the* | *bombing*) i.e. *the* conditioned on *bombing*, from backward network,

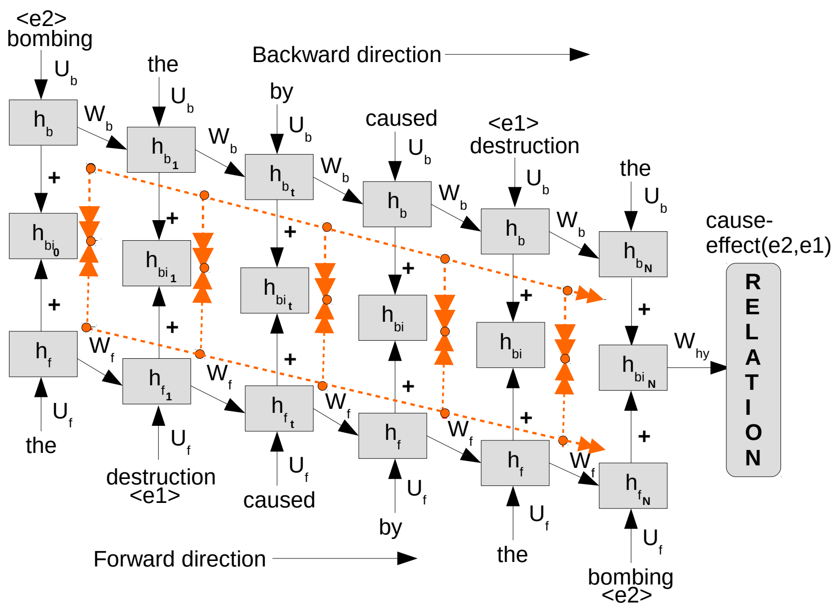

Figure 4.7: Bi-directional RNN with independent weights for relation classification (no weighted combination of forward and backward hidden units).

Therefore, h_{bi_1} has not accumulated the semantics for remaining words i.e. {*caused, by*}. But, as we propagate further in the network, the full information can be exploited, exactly after half of the sequence. Here, even though, the concatenated units, h_{bi}, don't utilize the full information, until half of the word sequence, but we still achieve the goal of bi-directionality, because, when supervision is performed at the end of the word sequence, all the information is being used at the time step, N, to update the network.

2. As in Fig 4.8, the hidden unit, at any time step t, from the backward network is combined with the forward unit that has the same input word. Since, any word input in the backward network is conditioned on the future words and words in the forward network are conditioned on the past words in the word sequence, therefore, all concatenated units, h_{bi}, utilize the full information (all words) throughout the word sequence. As in Fig 4.8, the orange colored arrows indicate the context flow in the BRNN. At any word input observed in forward network, for example, *'destruction'*, the corresponding h_{bi}, obtains context representation of -

   - (*destruction* | *the*) i.e. *destruction* conditioned on *the*, from forward network,

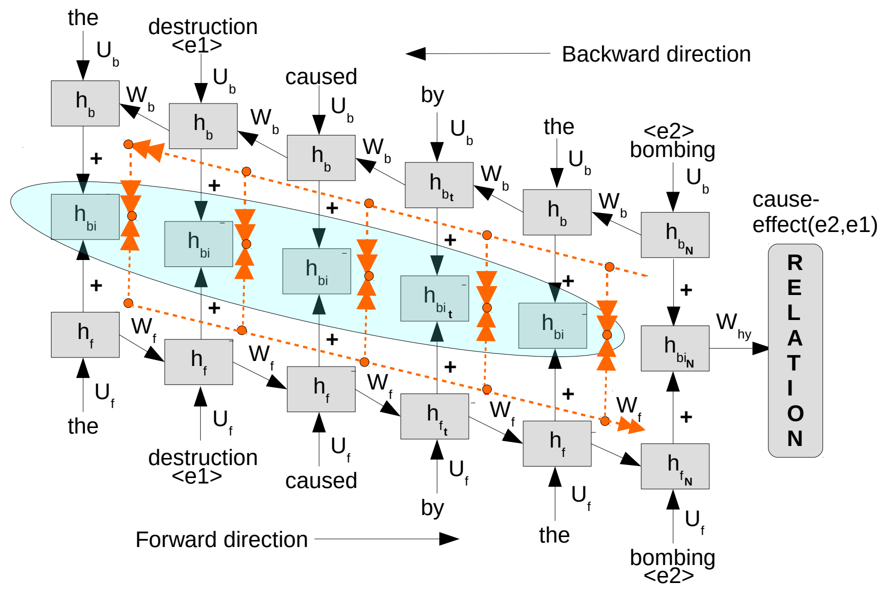

Figure 4.8: Bi-directional RNN with independent weights for relation classification (context flow demonstration).

- (*destruction* | (*caused* | (*by* | (*the* | *bombing*)))) i.e. *destruction* conditioned subsequently on {*caused, by, the, bombing*}, from the backward network.

Therefore, the full information is utilized at each concatenated unit and the context accumulated at $h_{bi} = (destruction|the) + (destruction|(caused|(by|(the|bombing))))$, where the current input word is *destruction* to the forward network.

One can also observe that the forward and backward networks are trained *jointly*, since the supervision is performed at the end of the sentence, after combining the forward and backward passes. We demonstrate the performance of these network for SemEval10 and slot filling datasets, and shall also discuss these two ways of combining could be interpreted as recursive semantic composition of patterns by recurrent networks.

Note, the bi-directional hidden units (h_{bi}) are not connected.

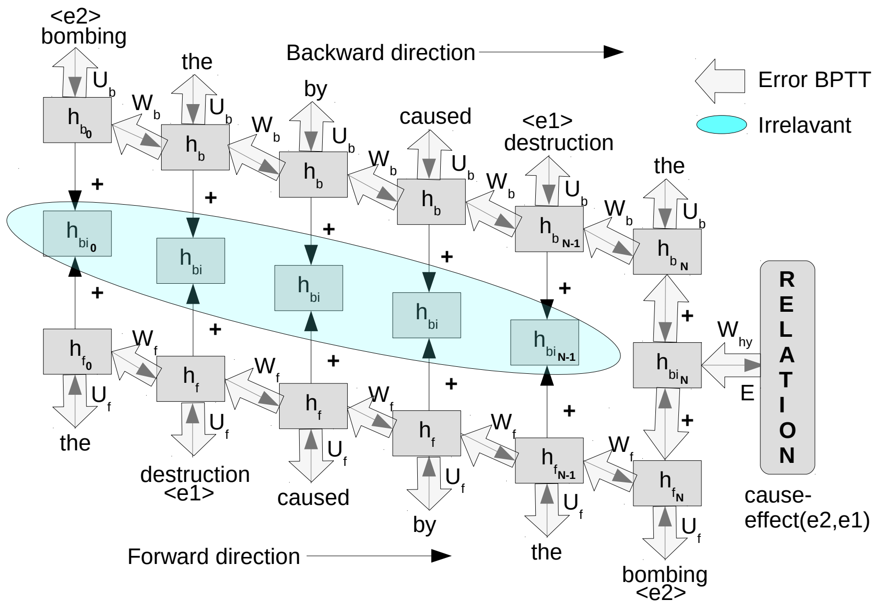

Figure 4.9: Backpropagation Through Time (BPTT) in standard BRNNs.

4.2.4 Backpropagation through time (BPTT)

We evaluated the performance of BRNN architectures (in Fig 4.7 and Fig 4.8) on SemEval10 and Slot Filling data sets. We found the performance of these models are similar to Uni-directional architectures for relation classification. Here, we investigated the limitation of these networks and observed that the concatenated hidden units (enclosed in colored oval shape), as in Fig 4.8, except the last concatenated unit h_{bi_N} are irrelevant. Since, the relation label is attached to the last concatenated hidden unit, h_{bi_N} and there are no time feedback connections among the concatenated hidden units, therefore, there is no error backpropagation from the relation label to the concatenated hidden units, except the last combined unit, h_{bi_N}. The Backpropagation Through Time (BPTT) is demonstrated in Fig 4.9 and the hidden units $(h_{bi0}, h_{bi1}, ..., h_{biN-1})$ does not contribute towards network learning.

Thus, the concatenated hidden units encapsulated in colored oval shape, as in Fig 4.9 and Fig 4.8, are irrelevant, if not connected through time. Therefore, these two networks look similar to two Uni-directional RNNs, combined and label attached at the end, and the full information (the past and future) is not utilized at each time step through the network.

4.2.5 Connectionist bi-directional RNNs (C-BRNN)

As we discussed the limitation of BRNN architectures in Fig 4.8 and Fig 4.9, here we proposed novel architectures, named as *Connectionist BRNNs* (C-BRNN), where the concatenated hidden units, h_{bi}, have time feedback connections among them. These connections allow to propagate the context accumulated from the combination of forward and backward network, at each time step from t_0 through t_N, at the end of the word sequence.

The semantics accumulated utilizing full information from the past and future at each concatenated hidden unit, h_{bi}, is propagated to the relation label and thus, each h_{bi} contributes to the relation learning. This introduction of connections among the concatenated hidden units leads to much superior results. Mathematically, the

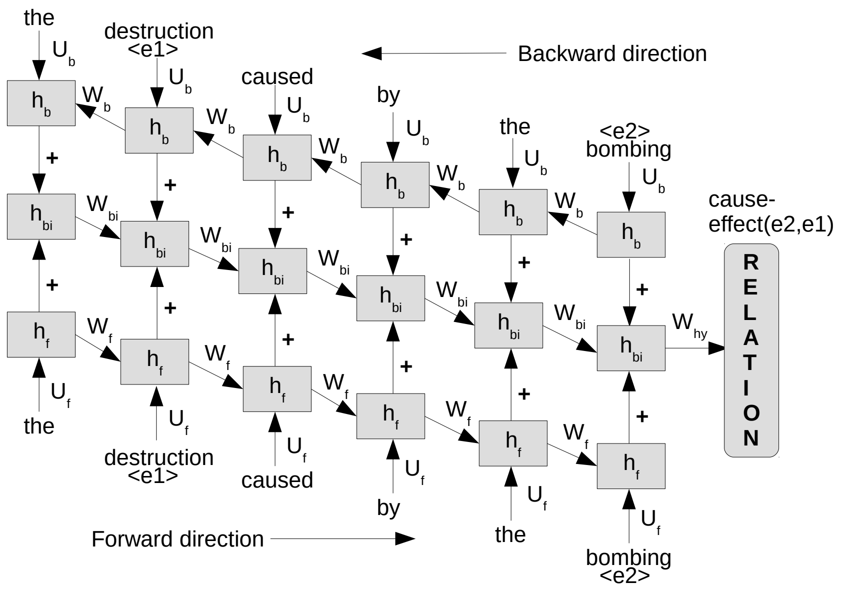

Figure 4.10: C-BRNN (variant 1) with independent weights for relation classification.

combined hidden units in *Connectionist BRNNs* (Fig 4.10, and Fig 4.11), are denoted as -

$$h_{bi_t} = f(h_{f_t} + h_{b_t} + W_{bi}h_{bi_{t-1}}) \tag{4.10}$$

$$h_{bi_N} = f(h_{f_N} + h_{b_N} + +W_{bi}h_{bi_{N-1}}) \tag{4.11}$$

where,

- W_{bi} : the weight matrix connecting combined hidden units, h_{bi} through time;

- $h_{bi_{t-1}}$: the concatenated hidden unit that accumulates the semantics obtained from the concatenation of forward and backward units at time step, $t-1$;
- N : the length of word sequence
- $h_{bi_{N-1}}$: the history accumulating the semantics from the forward, backward as well as concatenated networks, at time step, $N-1$.

Other notations and computations remains the same, as in Eq. 4.15, and 4.16.

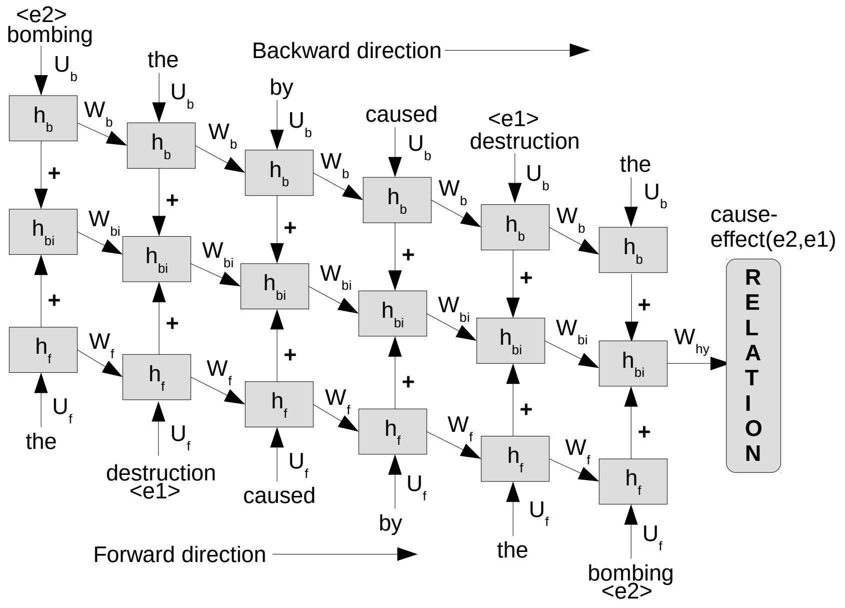

Figure 4.11: C-BRNN (variant 2) with independent weights for relation classification.

As we already discussed the two ways of combining the forward and backward networks, we also investigated the performance of the C-BRNN architectures, in Fig 4.10 and 4.11, where one could observe the difference in order of backward units used in combination with the forward units to obtain the hidden units, h_{bi}. There two variants of C-BRNNs are observed to perform similar on SemEval10 data set. We analysed these two ways of combining in C-BRNN network, (Fig 4.10 and 4.11), and observed that they could also be interpreted as recursively learning the semantic compositions of sub-patterns and propagation in connectionist framework, as described later in Fig 4.15. Observe that the bi-directional networks we discussed so far, use *independent weights (W_f and W_b) in forward and backward passes*. We shall investigate the impact of *shared weights* in forward and backward passes, instead of being independent.

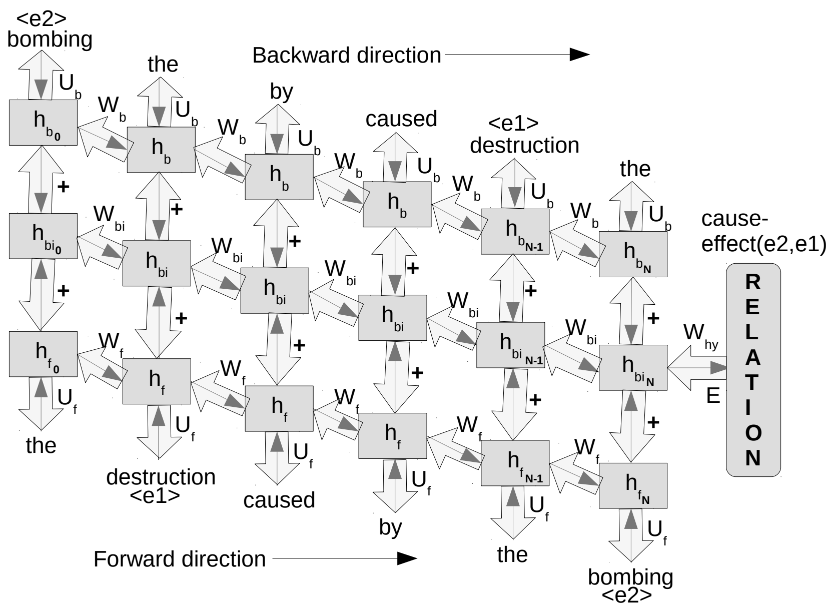

Figure 4.12: Backpropagation Through Time (BPTT) in Connectionist-BRNNs.

Backpropagation through time (BPTT) in C-BRNNs

In Fig 4.12, we demonstrate the error backpropagation through time in the Connectionist-BRNN. In contrast to Fig 4.9, we observe that the error is backpropagated from last concatenated hidden unit (h_{bi_N}) to initial concatenated hidden unit (h_{bi_0}) and weights (W_{bi}) are updated through the time steps, which allow the contribution of all the concatenated hidden units (h_{bi}) in the relation learning task.

The C-BRNN contributes to one of the novelty components that we bring to our thesis work and we also achieve the state-of-art performance on SemEval10 data set for relation classification task via C-RNNs.

4.2.6 Weight sharing strategies in forward/backward networks

The bi-directional networks discussed so far, used independent weights (W_f and W_b) in forward and backward passes. We also investigated the performance of bi-directional recurrent networks, where the weights (W_b) in backward network is shared with weights in forward network.

There are two weight sharing strategies, investigated on SemEval10 data set, as -

1. *SHARED* : The weights connecting hidden units in forward and backward networks are shared i.e. $W_f = W_b$, as stated in Fig 4.13. It leads to reducing

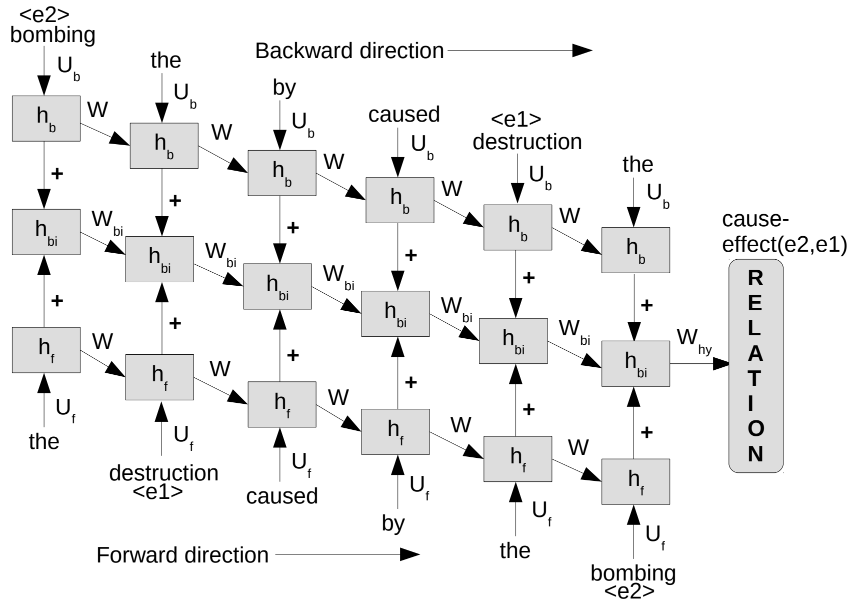

Figure 4.13: C-BRNN with shared weights for relation classification.

the number of free parameters as well as parameter search space, since they are shared in both networks throughout the word sequence. Therefore, the forward and backward units are obtained as -

$$h_{f_t} = f(U_f x_t + W_{shared} h_{f_{t-1}}) \tag{4.12}$$

$$h_{b_t} = f(U_b x_t + W_{shared} h_{b_{t+1}}) \tag{4.13}$$

where; W_{shared} is the shared weight matrix connecting the forward (h_f) and backward (h_b) hidden units in forward and backward networks, respectively.

2. $SHARED - TRANSPOSE$: Another strategy of weight sharing, we find intuitive to include for relation classification via BRNNs. Since, the input word sequence to backward network is reversed in order to compute to backward units (h_b) and is thought be a Uni-directional RNN in backward direction. Therefore, the weights (W) connecting forward hidden units is transposed and used in backward network to compute the backward units from reversed input word sequence, as mentioned in Fig 4.14. Since the weights are shared, but transposed therefore, the strategy is named as *SHARED-TRANSPOSE*.

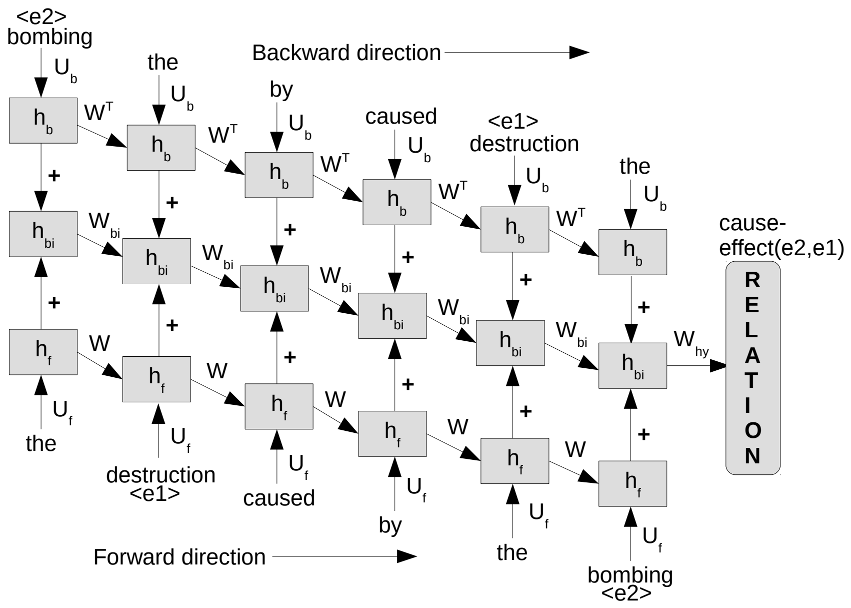

Figure 4.14: C-BRNN with shared-transpose weights for relation classification.

Mathematically, the forward and backward units are obtained as -

$$h_{f_t} = f(U_f x_t + W_{shared} h_{f_{t-1}}) \tag{4.14}$$

$$h_{b_t} = f(U_b x_t + W_{shared}^T h_{b_{t+1}}) \tag{4.15}$$

One advantage of the shared matrices is the *moderate number of free parameters* (weights), which reduces the risk of over-fitting [**Zimmermann2001**]. We observed pretty similar performance with *independent, shared and shared-transpose* weight strategies applied to C-RNN for relation classification on SemEval10 data set.

4.2.7 Semantic composition-accumulation-propagation in C-BRNN

As we discussed, the two ways of composition of forward and backward units in bi-directional recurrent networks based on the order of backward hidden units used in concatenation with forward hidden units. We demonstrate the two ways graphically in Fig 4.15. Here, we also observe that the connectionist network (C-BRNN) recursively performs composition of semantic meaning accumulated through the forward and backward networks, propagates the composition ultimately to the relation label via concatenated hidden units, (h_{bi}). Therefore, the term, *semantic composition-accumulation-propagation*.

Here, we briefly discuss the difference in the two ways of combining the network and reason why their performance is similar, on SemEval dataset. Further investigation on these semantic composition via BRNN is left to future work and is not in the scope of this thesis.

The semantic representation captured in first strategy (left in Fig 4.15) to some extent, could actually be seen as a subset of the semantic representation captured by second strategy (right in Fig 4.15), because we can observe that the semantics accumulated, in Fig 4.15 (Left), are also captured in Fig 4.15 (Right), as shown by the colored boxes. Since, these compositions obtained at each time step, are propagated through the network ultimately to the output layer, so we can say that we achieve the ultimate goal of bi-directionality (using the complete information, the past and future, from the input word sequence at each classification step i.e. the last time step in our case). As in Fig 4.12, the complete semantic meaning is accumulated word-by-word ultimately at the last concatenated hidden units (h_{bi_N}) and it is utilized for the relation learning (i.e. sentence-level gradients). Therefore, semantic compositions (patterns) are learned via *Connectionist BRNN* without any NLP tool.

Even, it is worth investigating that a relation label (given for a sentence) could be redundantly assigned to all the words in input sequence, classification at each bi-directional hidden units (h_{bi}), and the network optimized at all time step like RNN for sequence classification task, in Fig 2.5 in connectionist-style. However, we keep this investigation for future works.

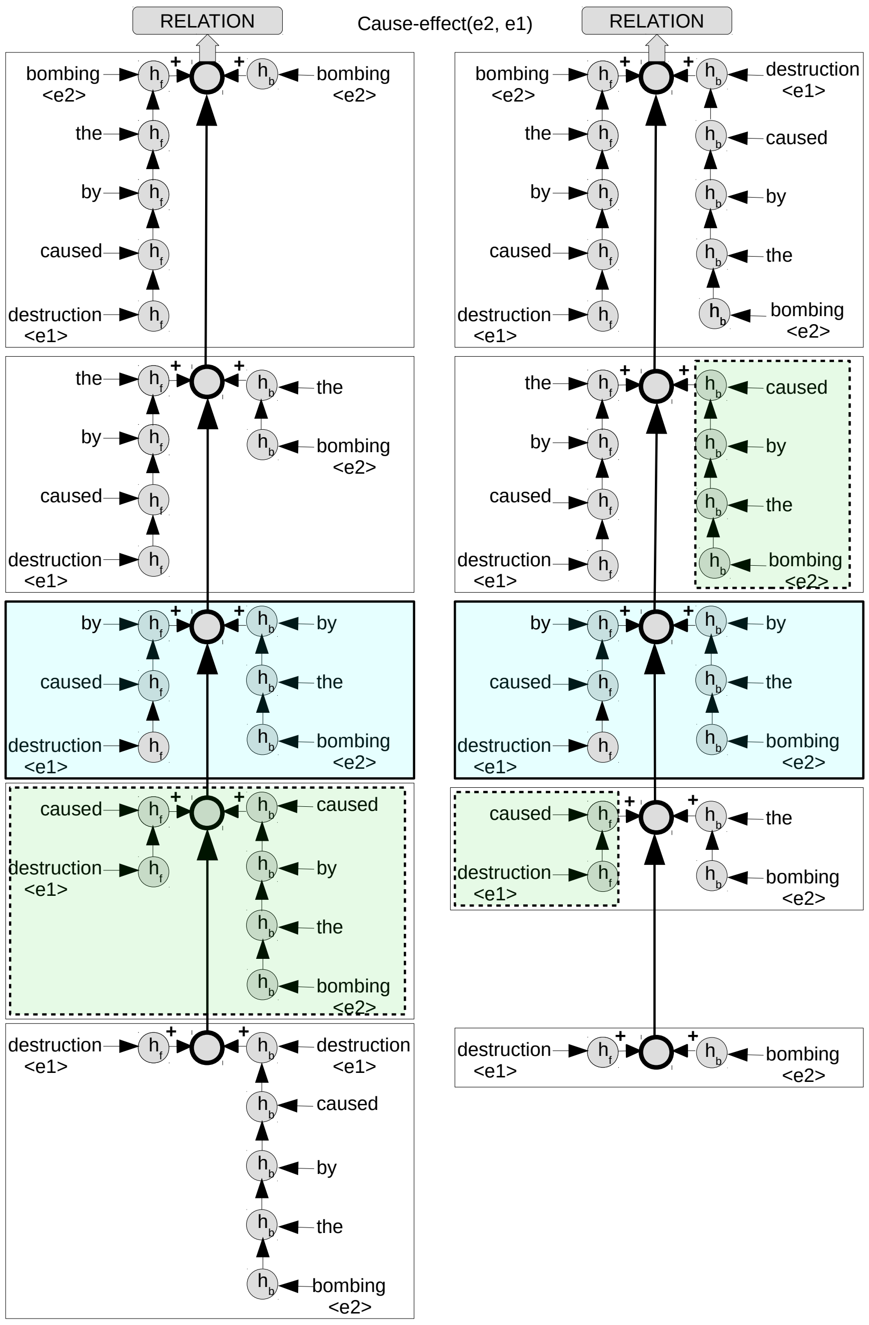

Figure 4.15: Recursive learning for semantic composition-accumulation-propagation of patterns in Connectionist-BRNN.

4.2.8 Connectionist multi-step-context BRNN (C-MSC BRNN)

To capture the long-term dependencies beyond the input window in the recurrent network, we investigated two different the multi-step RNNs architectures. Learning long-term dependencies with RNNs raises an optimization problem known as the *vanishing gradient problem*. Indeed, capturing longer time dependencies correspond to training a deeper model when the RNN is unfolded in time. Rather than training classical RNNs, we can directly provide some of the past information from different time steps or averaging in a window of hidden units, at each time step, as described below:

1. **Connectionist Multi-step BRNN window-averaged hidden units**: First, forward and backward hidden units are obtained via uni-directional propagation as discussed in BRNNs. The hidden units each in forward and backward network in a window of size (e.g. $w = 3$) are averaged and then the averaged units from forward and backward passes are combined.

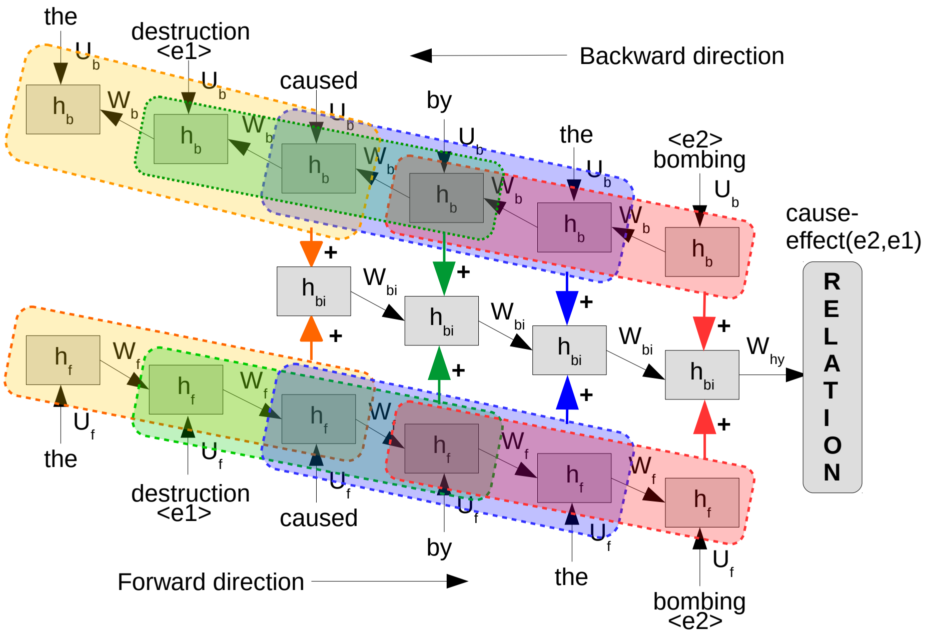

Figure 4.16: Connectionist Multi-step-context Bi-directional RNN (C-MSC bi-RNN) window-averaged hidden unitswith context step size=3.

Therefore, mathematically, the *Connectionist Multi-step RNN with window-average* *hidden units* as in Fig, at time-step t is given as,

$$h_{bi_t} = f(\frac{\sum_{i=0}^{w-1}(h_{f_{t-i}} + h_{b_{t-i}})}{w} + W_{h_{bi}h_{bi_{t-1}}}) \quad (4.16)$$

where;
w is the window size. Therefore, instead of relying on the current hidden unit, we obtain an average on w=3, hidden units to capture the longer context. Here, the context is learning in one step recurrence each in forward and backward network, while the multi-step is introduced at the moment of feeding the forward and backward networks to h_{bi}, the combined hidden unit. In Fig 4.16 , for example the orange colored box is the average of the three hidden units each from forward and backward network, and these two orange colored boxes are combined to obtained h_{bi}.

2. **Connectionist Multi-step BRNN with multi-timestep-connection feedback**: Instead of relying only on learning through one-step recurrences to capture context, one can combine recurrence with the idea of input window. This is achieved by

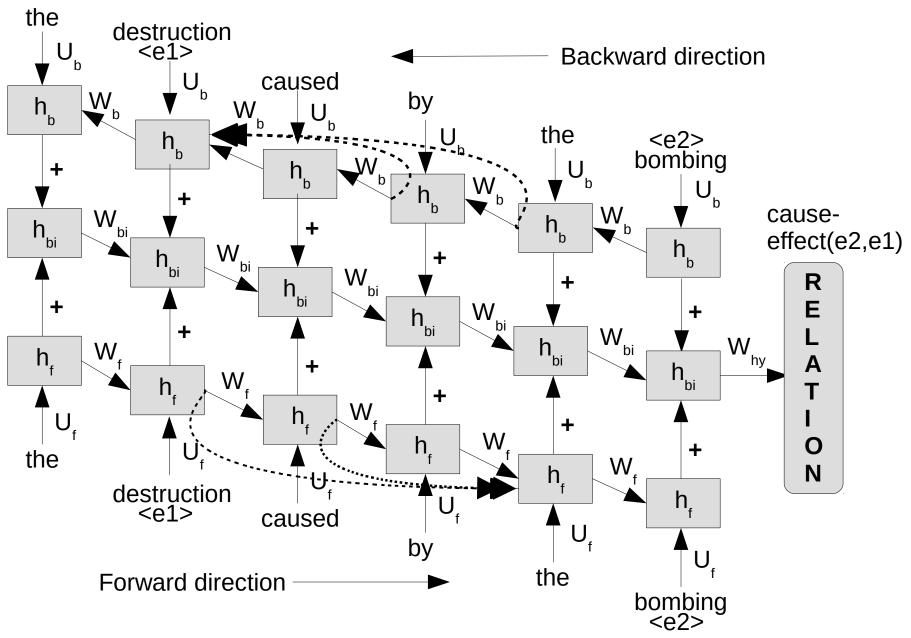

Figure 4.17: Connectionist Multi-step-context Bi-directional RNN (C-MSC BRNN) for relation classification with multi-timestep-connection feedback (w=3)

feeding the network with combination of the w previous time steps vectors, as mentioned in Fig 4.17, which is in contrast to Fig 4.16, where multi-step recurrence is achieved at the forward and backward network combination, not within the forward and backward networks itself to compute h_f and h_b respectively. The w previous step is again obtained by validation during training. Here, in our work, we use $w = 5$ previous time steps. Similar idea is also investigated by [**Mensil2013**] for sequence classification task in Jordon-style RNN networks. It also provides a way to obtain the most accurate number n-grams to consider for each relation types in relation classification task, which is also worth investigating in future.
Mathematically,
The forward hidden unit at time step, t, is obtained as -

$$h_{f_t} = f(W_{f_{xh}} x_t + \sum_{i=1}^{w} W_f h_{f_{t-i}}) \tag{4.17}$$

while, the input word sequence is reversed for the backward model, therefore, the hidden units in backward network represents the reversed context (i.e. h_b captures the context (past words) conditioned on the future words in word sequence). Therefore, the hidden unit at time step t in backward model for multi-step RNN is given by -

$$h_{b_t} = f(W_{b_{xh}} x_t + \sum_{i=1}^{w} W_b h_{b_{t-i}}) \tag{4.18}$$

In literature, the backward hidden units is also represented as -

$$h_{b_t} = f(W_{b_{xh}} x_t + \sum_{i=1}^{w} W_b h_{b_{t+i}}) \tag{4.19}$$

where '+' indicates the future context, which is similar to Eq 4.18 , where the input to backward network is reversed and a forward propagation is made to compute the backward units h_b.

We shall also compare the Multi-step BRNN (Fig 4.17 and 4.16) with a single word input and BRNN (Fig 4.11) with word features (N-grams) as input for SemEval dataset, where the number of step, w, in C-MSC BRNN is set equal to the number of words (w-Grams) as input in BRNN(Fig 4.11). Here, we find that training C-MSC BRNN is faster since, the input vector size is smaller. We show the performance of all the variants of bidirectional-RNNs as discussed here, in experimental section.

4.3 Ranking-RNN (R-RNN)

In some cases, it is important to learn for relevant classes and ignore the irrelevant ones or to reduce the impact of artificial classes. Since, in SemEval10 Task 8 dataset for relation classification, class *other* indicates that the samples does not belong to any of the remaining classes. Even, no similar pattern can be observed in the samples that belong to *other* classes. In the framework of CNN, [**Santos2015**] used pairwise ranking loss function in CR-CNN.

Therefore, we introduce a ranking loss function instead of softmax in recurrent neural network. We discussed in detail the ranking objective function in section 3.1.2.

Ranking offers the advantage of faster model training, in comparison to softmax, since during training at each round, two classes are updated i.e. the true class and the negative class. In this way, we can efficiently train the network for tasks which have a very large number of classes. In our experiments in R-RNN, given a sentence x with class label y^+, the incorrect class c^- that we choose to perform SGD step is the one with the highest score among all incorrect classes, therefore,

$$c^- = \arg \max_{c \epsilon C; c \neq y^+} s_\theta(x)_c \tag{4.20}$$

In our experiment for relation classification task on SemEval10 Task 10, we consider *Other* class as *artificial* since it groups items that do not belong to any of the actual classes and therefore, indicates that the relation between two nominals does not belong to any of the nine relation classes of interest. Hence, the class *Other* is very noisy since it groups many different types of relations that may not have much in common. During training, the first term in the right side of the ranking loss function becomes zero for a sentence x whose class label $y = Other$. Therefore, model learning for *natural* classes. During prediction time, a relation is classified as *Other* only if all actual classes have negative scores. Otherwise, it is classified with the class which has the largest score.

4.4 Combining RNN and CNN

Finally, we combine pre-existing CNN [**Adel2016**] and our RNN models. Our intuition is that their different way of sentence processing (convolution and max pooling vs. word-by-word processing for semantic accumulation of meaning) can lead to performance gain through combination.

Since ,we deal with two different data sets, Slot Filling and SemEval10. We used different strategies to combine the outputs of CNN and RNN models.

- **CNN-RNN combination for SemEval10 Task 8 dataset**: The combination is performed with a |*voting* process. For each sentence in the test set, we apply several CNN and RNN models, and take that class which gets the most votes by them. In case of a tie, we pick one of the most frequent classes randomly. This combination of these two different types neural networks achieves the state-of-art performance for relation classification on SemEval10 Task 8 dataset, which confirms our assumption that the networks provide complementary information.
- **CNN-RNN combination for Slot Filling dataset**: The combination is performed via *weighted combinations and setting a threshold*. Weights for each model (CNN and RNN) and the threshold on the combination scores are computed for each slot. The weights indicates the contribution of each model towards the combination scores, while threshold for each slot indicates the minimum confidence score required to label the sentence positive for that slot. This combination strategy is developed by [**Adel2016**] for the slot filling system, where multiple classifiers like SVM and CNN is used in combination with pattern matching approaches.

  Step in computing weights for each model and threshold for each slot in weighted combination of models:

  1. Compute weighted combinations of prediction probabilities (of being positive) for each slot from RNN and CNN.
  2. To compute the optimal threshold for each slot, compare each weighted combination scores against the list of thresholds.
  3. Results marked positive if the weighted combination scores is greater than the current threshold , t, in step 2.
  4. Compute F1 scores with true labels and the result obtained in step 3 for each slot for the threshold t.
  5. Get the optimal weights for each model and the corresponding optimal threshold for which the F1 score is the highest.

5 TAC KBP slot filling

The Text Analysis Conference (TAC) is a series of evaluation workshops organized to encourage research in Natural Language Processing and related applications, by providing a large test collection, common evaluation procedures, and a forum for organizations to share their results. The goal of TAC Knowledge Base Population (KBP) is to develop and evaluate technologies for populating knowledge bases (KBs) from unstructured text. We participate in the 2015 TAC KBP Cold Start English Slot Filling (SF) subtask, where we receive the evaluation queries, and produce only those entities and relations that would be found by the queries. The subtask extracts the values of specified attributes (i.e. slots) for a given entity from large collections of natural language texts. Examples of slots include age, birthplace, and spouse for a person or founder, top members, and website for organizations.

Our contribution for the 2015 TAC KBP research challenge is in the slot filler classification component using RNN, integrated into the slot filling pipeline (Fig 5.1) developed by CIS LMU [**Adel2014**].

5.1 Overview

The TAC KBP slot filling task addresses the challenge of gathering information about entities (persons or organizations) from a large amount of unstructured text data. We briefly discuss the slot filling system and its basic components, developed by [**Adel2014**].

5.2 Slot filling pipeline

The SF system [**Adel2014**] addresses the slot filling task in a modular way and is query driven. It has several advantages, including extensibility, analyzability of the different components and modular development. Fig 5.1 shows the components of SF pipeline. The following steps are performed, as explained by [**Adel2014**], in order to gather information about a person or organization and fill the pre-defined slots of the shared task -

- creation of a collection of possible aliases for the given name **(alias component)**
- retrieval of documents containing mentions of the entity **(information retrieval component)**
- retrieval of sentences with mentions of the entity and possible slot fillers **(candidate extraction component)**
- classification of the candidates **(slot filler classification component)**
- postprocessing of the candidates **(postprocessing component)**

The major contribution of our thesis work lies in the *Slot Filler Classification Component*, as in Fig 5.1, where we integrate our Uni-directional and Bi-directional recurrent neural networks for relation classification on the candidate sentences with possible slot fillers and with target entities tags as <name> and <filler>. We investigated the variants of RNNs discussed in chapter 4 for relation classification task on SF dataset.

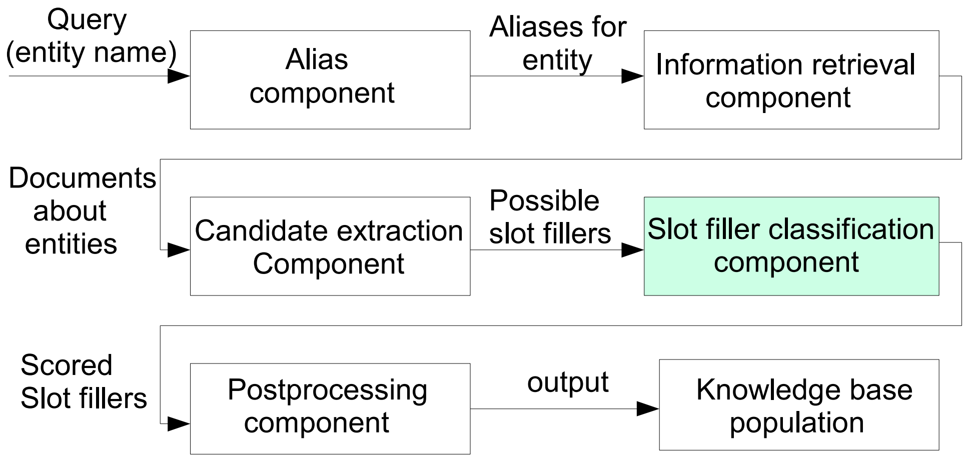

Figure 5.1: Basic Components of Slot Filling Pipeline by [**Adel2014**]

5.3 Recurrent neural networks in slot filling

Here, we investigate the potential of recurrent neural network for relation classification on the SF dataset. To the best of our knowledge, it is the first attempt to explore RNNs for slot filling task. Since, the SF task has candidate instances where the context length is usually high (Table 5.8), therefore, RNNs which are temporal models and can capture long term dependencies word-by-word, are worth investigating for SF task. We investigate various Uni-directional and Bi-directional RNN architectures, as discussed in chapter 4 and demonstrate the performance of RNN vs CNN based on the context lengths for slots *per:spouse* and *per:location_of_birth*. Note that in the SF dataset, we have multiple instances of nominals i.e. <name> and <filler> tags in the same sentence, as compared to SemEval10 Task 8 dataset where there are only 2 relation arguments for each sentence. The presence of multiple tags in SF dataset increases the complexity of the relation classification task.

5.4 Experimental setup

The variants of RNNs discussed in chapter 4 are implemented in Theano. In addition, the scripts to combine various classifiers in the *Slot Filler Classification Component* of the SF pipeline and to generate weights and thresholds for each model, are also implemented in python.

The SF system is evaluated using RNNs and the performance is compared to the other classifiers such as SVM and CNNs. We also combine multiple classifiers such RNNs and existing CNNs [**Adel2016**] along with SVM and pattern-based approach to evaluate the SF system, with an intuition that they have different way of sentence processing (convolution and max pooling vs. word-by-word processing and recurrence of hidden layer), which led to performance gains through combination.

In the first experiment, we evaluate the performance of RNN models using SF 2012+2013 data as development and SF 2014 as testing data to compare with state-of-the-art performance. Henceforth, we made a submission in research challenge the 2015 TAC KBP SF track [**TACKBP2015**] using RNN models integrated into SF pipeline by CIS LMU [**Adel2016**]. Here, we use SF 2012+2013+2014 as development dataset, while evaluation queries from the challenge are used as evaluation dataset. The major challenge with the task is to handle a number of models i.e. training a model for each slot (22 slots), and the optimization is tricky.

5.4.1 Data and evaluation metric

[**Surdeanu2014**] presents the list of slots for TAC KBP 2014 SF evaluation. The slot types can be: Name, i.e., named entities such as person, organizations, or locations; Value, i.e., numeric entities such as dates or other numbers; and String, which do not fall in any of the previous two categories. Further, an overview of the English Slot Filling Track at the TAC2014 Knowledge Base Population Evaluation is given by [**Surdeanu2014**]. Table 5.1 lists the slots we considered for SF evaluation in this thesis

Person Slots	Organization Slots
per:age	org:alternate_names
per:alternate_names	org:date_founded
per:children	org:founded_by
per:cause_of_death	org:location_of_headquarters
per:date_of_birth	org:members
per:date_of_death	org:parents
per:employee_or_member_of	org:top_members_employees
per:location_of_birth	
per:location_of_death	
per:locations_of_residence	
per:origin	
per:schools_attended	
per:siblings	
per:spouse	
per:title	

Table 5.1: List of slots for Slot Filling evaluation

work. There are 22 slots [**Surdeanu2014**] and each slot has a trained model. The 22 slots cover 57 of 67 slots because of the inverse slots (one model for two slots) and merging of location slots (city, state-or-province and country slots merged into one location slot for classification and are disambiguated again after classification).

The data is provided by CIS LMU [**Adel2016**], since the RNN models are integrated into SF pipeline and rely on the candidate slot fillers generated by the *candidate extraction component* (Fig 5.1). We use the standard micro F1 scoring metric.

5.5 Experimental results

We train individual model for each slot for the candidate sentences. We first evaluated our RNN models on TAC KBP SF 2014 queries to compare with the state-of-the-art. We take TAC KBP SF 2012+2013 dataset as development, while TAC KBP SF 2014 queries as evaluation set. For RNN training, we used SF data provided by [**Adel2014**] and prepared using distant supervision method [**AngeliandGupta2014**]. As discussed, RNN model is integrated into the existing slot filling system [**Adel2014**] and is evaluated against various classifiers, specially CNNs [**Adel2016**].

TAC KBP Slot Filling (DEV: 2012+2013, EVAL: 2014) position embeddings[+ entity flag] vs position indicators						
slots	Uni-RNN					
	position emb		position emb + entity flag		position ind	
	DEV	EVAL	DEV	EVAL	DEV	EVAL
per:age	.84	.69	.83	.71	**.86**	**.71**
per:alternate_names	.34	.04	**.36**	.06	.34	**.07**
per:children	.71	.42	**.73**	.38	.70	**.47**
per:cause_of_death	.74	.39	.72	**.48**	**.77**	.46
per:date_of_birth	1.0	.71	1.0	.77	1.0	.77
per:date_of_death	.71	.45	.71	.48	**.72**	**.55**
per:location_of_death	**.64**	.22	.62	.38	.63	**.43**
per:siblings	.66	.67	.67	.62	**.71**	**.70**
org:alternate_names	.58	.49	.59	.51	**.60**	**.55**
org:date_founded	.68	.66	.70	.67	**.80**	**.69**
average	.69	.47	.69	.51	**.71**	**.54**
median	.69	.47	.70	.49	**.72**	**.55**

Table 5.2: Uni-RNNs (position embeddings vs indicators): F1 scores on Slot Filling data set (DEV: 2012+2013, EVAL: 2014)

5.5.1 Uni- vs bi-directional RNNs

We first evaluate uni-directional RNNs with position features (position embeddings along with entity flags, Fig 4.2 and position indicators, Fig 4.3) on TAC KBP 2012+2013 as development, and TAC KBP 2014 as evaluation set. From Table 5.2, we observed that the position indicators (PI) lead to superior results as compared to position embeddings. Considerable improvement using PI is observed for slot *org:date_founded*. In addition, average F1 score is 0.71 and 0.54 for development and evaluation sets respectively, while it is 0.69 and 0.51 with position embeddings and entity flag combination, as detailed in the Table 5.2. Therefore, we further investigate BRNNs using PI features, instead of position embeddings and entity flags combination.

In Table 5.3, we present the performance of bi-directional RNNs for each slot and compare them Uni-directional RNN using PI. Here, we explored the C-BRNNs (Fig 4.11)

TAC KBP Slot Filling (DEV: 2012+2013, EVAL: 2014)								
slots	Uni-RNN		Connectionist BRNN (C-BRNN)					
			w_{hh} : *shared*		w_{hh} : *transpose*		w_{hh} : *independent*	
	DEV	EVAL	DEV	EVAL	DEV	EVAL	DEV	EVAL
per:age	.86	.71	.84	.71	.83	.69	**.86**	**.74**
per:alternate_names	.34	.07	.34	.07	.35	.06	**.38**	**.07**
per:children	.70	.47	**.76**	**.51**	.54	.31	.71	.51
per:cause_of_death	.77	.46	.77	.46	.77	.47	**.77**	**.52**
per:date_of_birth	1.0	.77	.99	.77	.95	.73	**1.0**	**.77**
per:date_of_death	.72	.55	.71	.57	.71	.53	**.71**	**.57**
per:employee_or_member_of	.43	.33	.42	.33	**.43**	**.37**	.39	.36
per:location_of_birth	.64	.31	**.67**	**.38**	.67	.37	.67	.32
per:location_of_death	.63	**.43**	.63	.28	**.65**	.34	.64	.34
per:locations_of_residence	.25	**.33**	.24	.29	**.28**	.29	.25	.29
per:origin	.41	.34	**.38**	**.32**	.32	.29	.36	.31
per:schools_attended	.67	.62	.69	.52	**.69**	**.63**	.61	.55
per:siblings	.71	.70	.70	.73	.71	.68	**.71**	**.73**
per:spouse	.71	.32	.74	.56	**.76**	**.61**	.74	.55
per:title	.41	.40	.40	.38	**.41**	**.41**	.31	.36
org:alternate_names	.60	.55	.59	.57	**.60**	**.58**	.59	.56
org:date_founded	.80	.69	.65	.64	.65	.70	**.66**	**.72**
org:founded_by	.78	.65	.77	.63	.68	.51	**.78**	**.68**
org:location_of_headquarters	.42	.43	.40	.43	.40	.44	**.40**	**.44**
org:members	.58	.21	.59	.09	.54	.18	**.59**	**.27**
org:parents	.33	.11	.25	.07	.28	.10	**.37**	**.11**
org:top_members_employees	.46	.48	.50	.50	.48	.51	**.50**	**.53**
average	.60	.45	.51	45	.58	.45	**.60**	**.47**
median	.63	.45	.64	.48	.63	.46	**.63**	**.52**

Table 5.3: RNNs+position indicators: F1 scores on Slot Filling data set (DEV: 2012+2013, EVAL: 2014)

models along with *independent*, *shared* (Fig 4.13) and *transposed* (Fig 4.14) weight strategies in forward/backward networks. Following the experiments in section 6.3.1 on SemEval10 Task 8 for relation classification, we observed that C-BRNN (Fig 4.11) is superior to other variant of BRNNs (Figures 4.7, 4.10, 4.16, 4.17), discussed in Chapter 4. Therefore, we investigate the capability of C-BRNNs for relation classification on SF dataset, using position indicators.

As shown in Table 5.3, we observe that C-BRNN is superior to uni-directional RNNs. The **bold** in Table 5.3 represents that the RNN model outperforms other RNN variant. For instance, we observe that slot *per:spouse* C-BRNN with *transposed* weights achieves the best F1 score of 0.76 and 0.61 on development and evaluation sets. While C-BRNN with *shared* and *independent* weight strategies scores similar for *per:spouse* slot. In comparison to uni-directional RNN with F1 scores of 0.71 and 0.32 on development and evaluation sets, C-BNN with *transposed* weights outperforms and achieves a considerable improvement on evaluation set i.e. +0.31. Similarly, for slot *per:location_of_birth*, the C-BRNN with *shared* weights leads to superior results as compared to other C-BNN with *independent* and *transposed* weights. Therefore, it is worth investigating bi-directional RNNs with three different weight sharing strategies, since either one outperforms the rest, as shown in Table 5.3.

The RNN model corresponding to the **bold** number under *DEV* column in Table 5.3 are chosen for comparison and combination with existing classifiers CNN [**Adel2016**], SVM and pattern-based approach [**Roth2013**] for slot filling. On selecting the top scoring RNN model from Table 5.3, we take these **bold** numbers into Table 5.4, where we compare the performance of existing models- pattern-based (PAT), SVM and CNN ([**Adel2016**]) with our RNN models. In Table 5.4, we first compare the performance of CNN with RNN and the **bold** numbers represents the RNN superior results for the respective slot. For instance, especially a large improvement for slot *per:spouse* is noticed, where CNN scores 0.67 and 0.30 while RNN achieves **0.76** and **0.61**, on the development and evaluation sets respectively. *Out of 22 slots, RNN in comparison to CNN leads to superior results for 10 slots*. Also, observe that the average F1 scores for *DEV*:**0.61** as well *EVAL*:**0.48** from RNN is slightly improved over CNNs (*DEV*:0.60 and *EVAL*:0.46).

The performance of CNN and RNN is further analysed for the slots *per:spouse* and *per:location_of_birth*, later in the section 5.6.1. So far, we evaluated our RNN models on TAC KBP SF-2014 evaluation dataset. Now, for the submission in the 2015 TAC KBP SF track, we train RNN models dataset provided by [**Adel2014**] (prepared using distant supervision method). Here, we used TAC KBP SF-2012+2013+2014 as development set. Using these trained RNN models, we submit the evaluation results for queries from the

TAC KBP Slot Filling (DEV: 2012+2013, EVAL: 2014)												
slots	PAT		SVM		CNN		CMB$_1$		RNN		CMB$_2$	CMB_2-CMB_1%
	DEV	EVAL	DEV	EVAL	DEV	EVAL	DEV	EVAL	DEV	EVAL	EVAL	EVAL
per:age	.69	.80	.84	.75	.83	.76	**.86**	.80	**.86**	.74	**.81**	+1
per:alternate_names	.50	.50	.35	.02	.32	.04	**.51**	.50	**.38**	**.07**	**.57**	+7
per:children	.10	.07	.71	.43	.82	.61	**.86**	.73	.76	.51	**.78**	+5
per:cause_of_death	.44	.11	**.80**	.36	.77	.52	.80	.31	.77	.52	**.56**	+25
per:date_of_birth	.67	.57	.99	.67	**1.0**	.77	**1.0**	.67	**1.0**	.77	**.91**	+24
per:date_of_death	.30	.32	.78	.50	.72	.48	**.79**	.55	.71	**.57**	**.69**	+14
per:employee_or_member_of	.24	.22	.43	.34	.41	.37	**.46**	.30	**.43**	.37	**.43**	+13
per:location_of_birth	.30	.30	.62	.29	.59	.23	**.75**	.37	**.67**	.38	**.46**	+9
per:location_of_death	.13	.00	.64	.27	.63	.28	**.67**	.34	.65	.34	**.40**	+6
per:locations_of_residence	.10	.03	**.28**	.27	.20	.23	**.28**	.27	**.28**	.29	**30**	+3
per:origin	.13	.11	**.68**	.59	.43	.39	**.68**	.59	.38	.32	**.60**	+1
per:schools_attended	.27	.26	**.78**	.67	.72	.55	.78	.67	.69	.63	**.70**	+3
per:siblings	.14	.50	.60	.64	.63	.70	.67	.67	**.71**	.73	**.78**	+11
per:spouse	.40	.53	.63	.29	.67	.30	.73	.57	**.76**	.61	**.74**	+17
per:title	.48	.42	.36	.30	.57	.46	**.58**	.46	.41	.41	**.48**	+2
org:alternate_names	.70	.71	.60	.63	.65	.66	**.72**	.65	.60	.58	**.72**	+7
org:date_founded	.47	.40	.61	.59	.64	.71	**.67**	.72	.66	.72	**.78**	+6
org:founded_by	.39	.62	**.85**	.76	.80	.68	**.85**	.72	.78	.68	**.77**	+5
org:location_of_headquarters	.39	.30	.43	.44	.43	.45	**.44**	.43	.40	.44	**.48**	+5
org:members	.03	.29	.49	.16	.65	.04	**.68**	.07	.59	.27	**.45**	+38
org:parents	.31	.18	.28	.10	.41	.16	**.50**	.18	.37	.11	**.38**	+20
org:top_members_employees	.53	.46	.51	.57	.43	.53	**.58**	.53	.50	.53	**.65**	+12
average	.35	.36	.59	.44	.60	.46	**.67**	.51	.61	.48	**.61**	**+10**
median	.32	.35	.61	.44	.64	.67	**.68**	.54	.66	.52	**.63**	

Table 5.4: RNNs: F1 scores on Slot Filling data set (DEV: 2012+2013, EVAL: 2014, CMB_1:(PAT+SVM+CNN), CMB_2:(PAT+SVM+CNN+RNN)). The highlights show that superior scores for DEV and EVAL respectively under their column.

TAC KBP Slot Filling (DEV: 2012+2013+2014)								
slots	PAT	SVM	CNN	CMB_1	Uni-RNN	BRNN*	CMB_2	CMB_2-CMB_1%
per:age	.17	.76	.81	.83	.82	.81	**.84**	+1
per:alternate_names	.20	.08	.11	.22	.12	.11	**.23**	+2
per:children	.38	.74	.73	.74	.74	.73	**.76**	+2
per:cause_of_death	.48	.42	.60	.65	.60	.62	**.68**	+3
per:date_of_birth	.0	.96	.98	.98	.97	.97	.98	+0
per:date_of_death	.0	.75	.71	.76	.72	.71	**.77**	+1
per:employee_or_member_of	.12	.23	.26	.28	.23	.23	**.30**	+2
per:location_of_birth	.3	.51	.54	.67	.58	.61	**.69**	+2
per:location_of_death	.11	.58	.58	.63	.63	.63	**.67**	+4
per:locations_of_residence	.29	.23	.23	.29	.23	.21	**.31**	+2
per:origin	.03	.30	.30	.32	.30	.30	**.33**	+1
per:schools_attended	.26	.65	.63	.65	.65	.62	**.67**	+2
per:siblings	.4	.53	.57	.65	.56	.57	**.67**	+2
per:spouse	.47	.31	.43	.61	.43	.48	**.63**	+2
per:title	.53	.34	.55	.56	.52	.53	**.58**	+2
org:alternate_names	.58	.27	.41	.59	.29	.32	.59	+0
org:date_founded	.48	.62	.68	.72	.72	.70	**.74**	+2
org:founded_by	.09	.57	.64	.62	.60	.61	**.65**	+3
org:location_of_headquarters	.24	.43	.43	.46	.44	.38	**.47**	+1
org:members	.0	.28	.42	.44	.43	.41	**.50**	+6
org:parents	.06	.14	.26	.30	.24	.21	**.35**	+5
org:top_members_employees	.38	.35	.32	.51	.29	.29	**.52**	+1
average	.25	.46	.51	.57	.51	.50	**.59**	**+2.1**
median	.25	.43	.55	.62	.55	.55	.64	

Table 5.5: F1 scores on Slot Filling data set (DEV: 2012+2013+2014, EVAL: 2015, CMB_1:(PAT + SVM + CNN) [**Adel2014**], CMB_2:(PAT + SVM + CNN + Uni-RNN + BRNN)). * indicates BRNN models partially optimized at the time of the 2015 TAC KBP SF submission

Models	P (%)	R (%)	F1 (%)
CIS PAT+SVM+CNN normal thresholds	12.7	30.1	17.9
CIS PAT+SVM+CNN higher thresholds ([**Adel2014**])	25.8	18.1	21.3
CIS PAT+SVM+CNN+RNN normal thresholds	16.3	**30.4**	21.2
CIS PAT+SVM+CNN+RNN higher thresholds	24.5	17.3	20.0
Stanford ([**Gupta2014**], [**Surdeanu**])	**54.6**	27.8	**36.8**

Table 5.6: The 2014 KBP SF evaluation results

research challenge.

5.5.2 Combining CNN and RNN

In Table 5.3, the CMB_1 represents the combination of models (PAT+SVM+CNN), while CMB_2 is the combination of models (PAT+SVM+CNN+**RNN**). We also report the CMB_1 and CMB_2 scores for each slot using TAC KBP SF-2012+2013 and SF-2014 as development and evaluation sets, respectively. The existing CNN for slot filling ([**Adel2016**]) and our RNN models are combined via a weighting strategy, as detailed in section 4.4. In Table 5.5, the integration of RNN models i.e. CMB_2 clearly results in improvements on development set as compared to CMB_1 without RNN models.

From Table 5.4, it is observed that the $COMB_2$ i.e. PAT+SVM+CNN+RNN leads to superior results as compared to $COMB_1$ i.e. PAT+SVM+CNN for all the slots in SF dataset. The average improvement is +10 on 2014 evaluation set. Here, the integration of RNN improves scores for individual slots. Thus, we adavantage from the different types of sentence representation from CNN and RNN, when combined.

5.5.3 State-of-the-art performance

Here, we compare the overall performance of the slot filling system (Fig 5.1) on integrating RNN models into *slot filler classification component*, developed at CIS LMU ([**Adel2014**]).

Here, in Table 5.6, we observe that the PAT+SVM+CNN+RNN combination improves the precision (P), recall (R), and F1 scores for normal threshold as compared to PAT+SVM+CNN combination, while, for higher thresholds PAT+SVM+CNN gains considerable overall system improvement. It indicates that the RNN does recognize the

Official Scoring	Models Submitted	hop	P (%)	R (%)	F1 (%)
CSLDC max	PAT+SVM+CNN ([**Adel2016**])	0	32.62	22.74	26.80
		1	13.73	7.25	9.49
		ALL	**27.18**	17.35	21.18
	PAT+SVM+CNN+RNN	0	29.98	24.23	26.80
		1	13.01	06.51	8.67
		ALL	25.76	**18.06**	**21.23**
CSSF	PAT+SVM+CNN ([**Adel2016**])	0	29.41	19.01	23.09
		1	13.85	6.94	9.24
		ALL	**24.55**	14.55	18.27
	PAT+SVM+CNN+RNN	0	27.05	21.42	23.91
		1	11.06	5.24	7.12
		ALL	22.89	**15.44**	**18.44**

Table 5.7: The 2015 KBP SF evaluation results (*preliminary*)

correct answers but does not give them the highest confidences. Here, we compare the two combination with/out RNNs integrated into CIS SF system. Table 5.6 also shows the state-of-art results from winners of the 2014 TAC KBP SF track.

Table 5.7, presents the 2015 TAC KBP Slot Filling *preliminary* scores published so far, because they are based on a (large) subset of the final set of queries that will be assessed and scored. There are two official evaluation metrics: CSLDC max and CSSF. We see from the table 5.7 that RNN in combination with PAT+SVM+CNN improves the overall (i.e. **ALL** column) scores. Similar to the 2014 KBP evaluation results in table 5.6, we observe that the PAT+SVM+CNN+RNN combination improves the recall as compared to PAT+SVM+CNN, as highlighted in table 5.7, but not precision. As observed, the RNN models in combination with pattern-based approach, SVM and CNN [**Adel2016**] helped improving the 2015 TAC KBP SF scores.

5.6 Analysis on RNN and CNN performance

Here, we analyze the performance of CNN and RNN based on context length. We also demonstrate the semantic meaning accumulation behavior by RNN through word-by-word at each time-step and ultimately, we present the most representative tri-grams detected from RNN model. The analysis is performed on two slot: *per:location_of_birth* and *per:spouse*. We chose the slots, since RNN gains considerable improvements for these slots as compared to CNN performance.

5.6.1 Sentence lengths: Impact of context length

This is intuitive that too small context involves limited semantic information, while too large context leads to difficulties in pattern learning. Therefore, the advantage of RNN training over CNN is that RNN can deal with long distance patterns more effectively by recursively learning word-by-word and propagating contribution of distance patterns into the semantic meaning of the sentence.

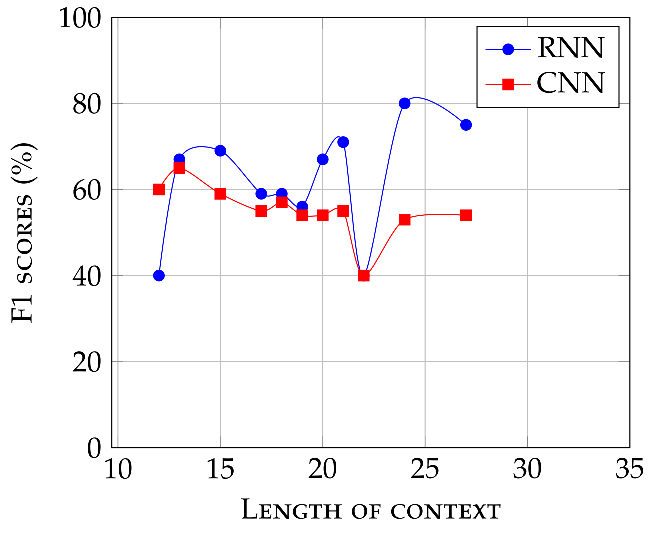

Figure 5.2: F1 scores of RNN and CNN with different context lengths for slot *per:location_of_birth*.

We investigate the performance of RNN and CNN models for slots *per:location_of_birth* and *per:spouse* based on input context length. The sentences in the development dataset of these slot are indexed by the context lengths. Then, we split the test dataset into the number of various test sentence lengths. Here, the context is defined as words

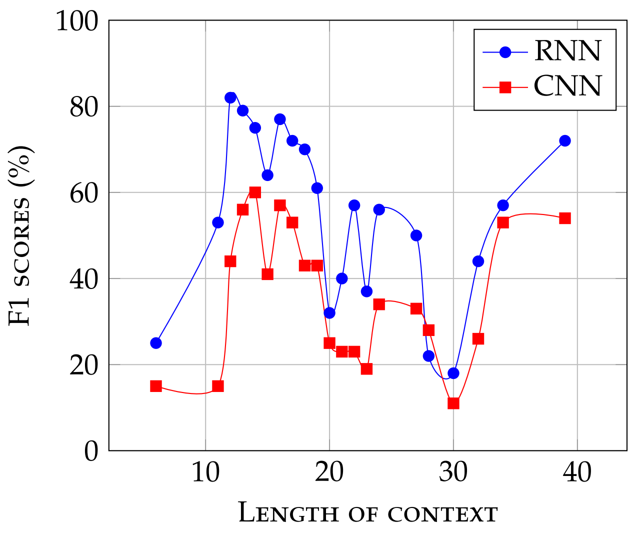

Figure 5.3: F1 scores of RNN and CNN with different context lengths for slot *per:spouse*.

Slots	Context Length	Number of sentences
per:location_of_birth	< 15	250
	15-20	231
	21-30	120
	> 30	27
per:spouse	<= 10	771
	11-15	710
	16-20	858
	21-30	**906**
	> 30	185

Table 5.8: The distribution of context length in SF development dataset for slots: *per:location_of_birth* and *per:spouse*.

between the two nominals (<name> and <filler>) plus extended context, i.e. 5 words prior to the first nominal and 5 words after the second nominal, if they exist. The context length also includes 4 additional independent words as the position indicators for the nominals. Both models have the same of samples to compute the F1 scores.

Clearly, long contexts lead to long-distance patterns. It can be seen in Fig 5.2 for slot *per:location_of_birth* that if the context length is small (<=12), the CNN performs better, whereas if the context length is large (>=20), the RNN model is clearly superior. While, in moderate context length (15-20), they are comparable. Similarly, for slot *per:spouse*, when the context length is higher (>=20), RNN clearly leads to CNN. This confirms that RNN is more suitable to learn long-distance patterns. Note that with both the two models, the best F1 results are obtained with a moderate length of contexts. This is understandable as too small context involves limited semantic information, while too large context leads to difficulties in pattern learning. The distribution of the context length in SF development set for these two slots are mentioned in the Table 5.8.

5.6.2 Semantic accumulation by C-BRNNs

RNNs are temporal models and are cumulative in nature. They accumulate the semantic meaning of a sentence word-by-word at each time point. While, CNN model learns only local patterns.

Here, we demonstrate the nature of recurrent neural networks for *per:location_of_birth* and *per:spouse* slots, using C-BRNN architecture, that captures the semantic meaning of sentence word-by-word. The peaks in the figures (Fig 5.4 and 5.5) indicate the importance of word or word segment (pattern) in representing the semantics of the sentence.

For instance, to compute the probabilities (on y-axis) in figures 5.4 and 5.5, we take our C-BRNN trained model and do prediction on the given input the word/word segments, as mentioned in Tables 5.9 and 5.10. Bold indicates the last word in word segment input to C-BRNN. and figures 5.4 and 5.5 demonstrate how we capture the semantic meaning accumulation nature of recurrent neural networks for given word sequence.

Input word sequence to C-BRNN	prediction probability
<e1>	0.34
<e1> **person**	0.34
<e1> person **</e1>**	0.34
<e1> person </e1> **was**	0.37
<e1> person </e1> was **born**	0.50
<e1> person </e1> was born **in**	0.58
<e1> person </e1> was born in **<e2>**	0.53
<e1> person </e1> was born in <e2> **location**	0.54
<e1> person </e1> was born in <e2> location **</e2>**	0.53
<e1> person </e1> was born in <e2> location </e2> **,**	0.69
<e1> person </e1> was born in <e2> location </e2> **maryland**	0.71
<e1> person </e1> was born in <e2> location </e2> maryland **,**	0.81
<e1> person </e1> was born in <e2> location </e2> maryland , **on**	0.80

Table 5.9: Semantic Meaning Accumulation by C-BRNN word-by-word for word segment from slot *per:location_of_birth*. Bold indicates the last word in word segment input to C-BRNN.

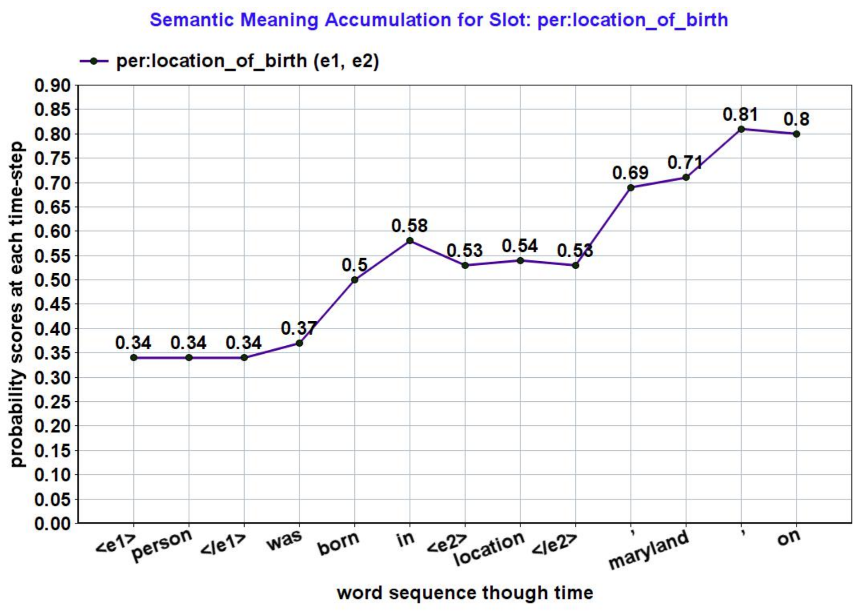

Figure 5.4: Semantic Meaning Accumulation by C-BRNNs for Slot: *per:location_of_birth*

Input word sequence to C-BRNN	prediction probability
a	0.02
a, **time**	0.02
a time **later**	0.02
a time later **<e1>**	0.02
a time later <e1> **person**	0.02
a time later <e1> person **</e1>**	0.10
a time later <e1> person </e1> **married**	0.98
a time later <e1> person </e1> married **<e2>**	0.99
a time later <e1> person </e1> married <e2> **spouse**	1.0
a time later <e1> person </e1> married <e2> spouse **</e2>**	0.98
a time later <e1> person </e1> married <e2> spouse </e2> **a**	0.93
a time later <e1> person </e1> married <e2> spouse </e2> a **daughter**	0.97
a time later <e1> person </e1> married <e2> spouse </e2> a daughter **of**	0.98

Table 5.10: Semantic Meaning Accumulation by C-BRNN word-by-word for word segment from slot *per:spouse*. Bold indicates the last word in word segment input to C-BRNN.

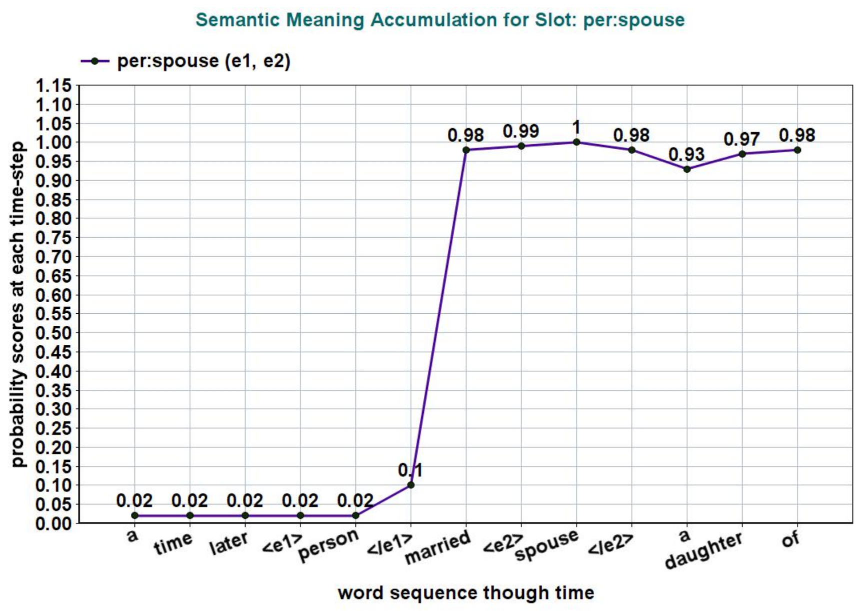

Figure 5.5: Semantic Meaning Accumulation by C-BRNNs for Slot: *per:spouse*

5.6.3 Representative n-grams detected

Slots	tri-gram Patterns Detected
per:spouse(e1, e2)	[</e1> wife of] [</e1> , wife] [</e1> ′ wife] [</e1> married <e2>] [</e1> marriages to]
per:location_of_birth(e1, e2)	[was born in] [born in <e2>] [a native of] [</e1> from <e2>] [</e1> ′ hometown]

Table 5.11: The Top 5 representative tri-grams detected for *per:spouse* slot and *per:location_of_birth* slot in SF dataset by C-BRNNs with PI

Here , we detect the most representative tri-grams from RNN models. To compute these tri-grams, we provide an input word segment generated from sliding a window of 3 words through the sentence. The probability scores are computed from each word segment and the most representative tri-grams are extracted, which contribute the most to the correct relation prediction. We have generated top 5 patterns for slots per:location_of_birth and per:spouse, as mentioned in Table 5.11 and these patterns could be used as seed for relation extraction tasks.

While detecting tri-gram patterns, we used position indicators as independent words in the sentences. Here, the position indicators stand for :

- <e1>: beginning of the slot word
- </e1>: end of the first slot word
- <e2>: beginning of the filler word
- </e2>: end of the filler word

6 Relation classification for SemEval10 dataset

6.1 Overview: SemEval task

Given a sentence and two annotated nominals, the Semantic Evaluation 2010 (SemEval10) task 8 predicts the semantic relations between the pairs of nominals. The most suitable relation label is assigned from the inventory of 19 relations, as defined in Table 6.1. For instance, a sentence with relation label is given by -

S: the <e1>bow</e1> of the <e2>vessel</e2> points at the extended edge of the table. $\rightarrow Component - Whole(e1, e2)$

The terms *bow* and *vessel* are the relation arguments or nominals and the focus of the SemEval task is to classify relation i.e. Component-Whole(e1,e2), between these pairs of nominals, using the context information given in the sentence.

SemEval10 Task 8 Semantic Relations	
Cause-Effect (e1, e2)	Cause-Effect (e2, e1)
Component-Whole (e1, e2)	Component-Whole (e1, e2)
Content-Container (e1, e2)	Content-Container (e2, e1)
Entity-Destination (e1, e2)	Entity-Destination (e2, e1)
Entity-Origin (e1, e2)	Entity-Origin (e2, e1)
Instrument-Agency (e1, e2)	Instrument-Agency (e2, e1)
Member-Collection (e1, e2)	Member-Collection (e2, e1)
Message-Topic (e1, e2)	Message-Topic (e2, e1)
Product-Producer (e1, e2)	Product-Producer (e2, e1)
Other	

Table 6.1: SemEval10 Task 8 semantic relations

6.2 Experimental setup

6.2.1 Data and evaluation metric

We use the dataset and evaluation framework provided by SemEval-2010 Task 8. There are 10,717 training and testing examples with 9 directional relations and an additional artificial relation, *'other'*, resulting in 19 relation classes in total. The *'other'* relation indicates that the relation in the example does not belong to any of the nine main relation types. The nine relations are *Cause-Effect, Component-Whole, Content-Container, Entity-Destination, Entity-Origin, Instrument-Agency, Member-Collection, Message-Topic and Product-Producer*. Each example contains a sentence marked with two nominals *e1* and *e2*, and the task consists of predicting the relation between the two nominals taking into consideration the directionality. That means that the relation *Cause-Effect(e1,e2)* is different from the relation *Cause-Effect(e2,e1)*, as shown in the examples below -

The <e1>**bombing**</e1> resulted in the <e**2>deaths**</e2> of 1318 in Hanoi.
$\Rightarrow Cause - Effect(e1, e2)$
The <e1>**flooding**</e1> has been caused by the <e2>**clogging**</e2> of drains.
$\Rightarrow Cause - Effect(e2, e1)$

The SemEval-2010 Task 8 dataset is already partitioned into 8,000 training instances and 2,717 test instances. Out of 8,000 instances, we take first 6500 for training and remaining 1500 as development set. We optimise on development set and predict the results for 2,717 testing instances. Given a sentence and two target nominals, a prediction is counted as correct only when both the relation and its direction are correct. The performance is evaluated in terms of the F1 score defined by SemEval-2010 Task 8 [**Hendrickx2009**]. We applied the official scoring script and report the macro F1 score which also served as the official result of the shared task.

6.3 Experimental results

6.3.1 Uni- vs bi-directional RNNs

We start our experiments on SemEval10 dataset using uni-directional RNN, as presented in Fig 4.1. Here, the RNN accumulates the semantic meaning of the whole sentence word-by-word at each time step using a single input word vector at each time-step. The relation label is attached to the last hidden unit where the supervision is performed and a softmax layer predicts the probabilities for each relation label. We use the F1 scores of 60.0 from uni-directional RNN as baseline for the experiments on the SemEval10 dataset.

Further, we investigate uni-direction RNN performance with different position features, such as position embeddings, position embeddings concatenated with a entity flags indicating if the current word is an entity mention, as depicted in Fig 4.2, and position indicators, in Fig 4.3. As discussed in section 4.1.1, position indicators (PI) are the independent words in the sentence indicating the entity presence in the sentence. We observed from Table 6.2 that PI features in uni-directional RNN lead to considerable improvement in F1 scores as compared both to position features and position features in combination with the entity flags. With position indicators in uni-directional RNN model, we achieve F1 scores of 61.9 and 73.4 on SemEval10 development and test sets, respectively. Instead of single word vector as input, we input tri-gram and 5-gram word vectors to uni-directional RNN along with increasing the word embedding dimensionality from 50 to 400 lead to improvement in the uni-directional RNN performance. As mentioned in Table 6.2, considering the PI, WF and 400 word embedding dimensions, the best score from uni-directional RNN is 77.6 on SemEval10 test dataset. In addition, we shuffle all training instances for each epoch in our experiments.

Following the performance of PI in uni-directional RNNs, we further investigate bi-directional RNNs (BRNNs) with PI. The BRNN (Fig 4.7 , with no time connection feedback among combined hidden units, h_{bi}) with PI results in 74.2, while a significant improvement is observed with our proposed model C-BRNN (Fig 4.11, with time connection feedback among h_{bi}) and scores 78.4. then, the word features (WF) N-gram (5-gram) improve the results to 79.9 with softmax output layer. Further, we used ranking layer, as discussed in R-RNN (section 4.3), instead of softmax output layer and it improves the F1 scores to 81.4. We observed that increasing the word embedding dimension to 400 results in 82.5 on test set. While [**Zeng2014**] use emb dim = 50 trained by [**Turian2010**] and [**Santos2015**] used emb dim = 400 trained with the word2vec toolkit [**Mikolov2013**]. The results in Table 6.2 show that increasing the word embedding dimension gains considerable improvement. We observed that the C-BRNN improves the F1 score on test set to 78.4, as compared to uni-directional RNNs score of 73.4 using only PI. Therefore, the bi-directionality achieves a large gain of 5.0 points.

We also investigate the different ways of treating the weight matrices i.e. *independent*, *shared* and *transposed* in the forward and backward network of the BRNN, as demonstrated in Figues 4.13 and 4.14. We observed that the shared and transposed weights strategies are comparable in performance to independent weight matrices in forward and backward networks of BRNNs for SemEval10 dataset.

We also investigated the two ways of combining forward and backward networks on

RNN models	Features	Dev	Eval
Uni-directional	emb dim = 50	52.6	60.0
	+ PF	57.8	68.3
	+ Entity flag	59.2	73.1
	PI	61.9	73.4
	+ WF (3-gram)	62.8	74.9
	+ WF (5-gram)	63.6	75.4
	+ increase emb dim to 400	66.2	77.6
BRNN, no time feedback connection between combined h_{bi}, Fig 4.7 + independent weights	PI	63.7	74.2
C-BRNN (Fig 4.11) + independent weights	PI	68.7	78.4
	+ WF (5-gram)	69.7	79.9
C-BRNN (Fig 4.11) + independent weights **(R-RNN)**	+ ranking	70.6	81.4
	+ increase emb dim to 400	**72.8**	**82.5**
C-BRNN (Fig 4.13) + shared weights	PI + WF (5-gram) + ranking + emb dim = 400	72.3	82.2
C-BRNN (Fig 4.14) + transpose weights	PI + WF (5-gram) + ranking + emb dim = 400	72.4	82.3
C-BRNN (Fig 4.10) + independent weights	PI + WF (5-gram) + ranking + emb dim = 400	72.3	81.6
C-BRNN (Fig 4.10) + shared weights	PI + WF (5-gram) + ranking + emb dim = 400	72.3	81.2
C-BRNN (Fig 4.10) + transposed weights	PI + WF (5-gram) + ranking + emb dim = 400	72.4	81.6
C-MSC BRNN, window averaged hidden units (Fig 4.16) + independent weights	PI + WF (5-gram) + ranking + emb dim = 400	71.1	81.0
C-MSC BRNN, window averaged hidden units (Fig 4.16) + shared weights	PI + WF (5-gram) + ranking + emb dim = 400	71.3	81.9
C-MSC BRNN, window averaged hidden units (Fig 4.16) + transposed weights	PI + WF (5-gram) + ranking + emb dim = 400	71.4	81.9
C-MSC BRNN, hidden units with multi-timestep connection feedback (Fig 4.17) + independent weights	PI + WF (5-gram) + ranking + emb dim = 400	71.7	81.9
C-MSC BRNN, hidden units with multi-timestep connection feedback (Fig 4.17) + shared weights	PI + WF (5-gram) + ranking + emb dim = 400	71.8	82.1
C-MSC BRNN, hidden units with multi-timestep connection feedback (Fig 4.17) + transposed weights	PI + WF (5-gram) + ranking + emb dim = 400	72.1	82.2

Table 6.2: F1 scores of RNN for relation classification on SemEval10 Task 8 dataset

BRNNs, as discussed in section 4.2.3. We observed that the combination presented in Fig 4.11 performs superior to the combination in Fig 4.10.

Further, we made some investigations on multi-step BRNNs with independent weight strategy. Here, we optimise the two variants of C-MSC BRNN (Fig 4.16 and Fig 4.17) using the same optimal parameter setting obtained for C-BRNN, in Table 6.4. We observed the performance comparable is to C-BNNs. Also, we keep further investigations on C-MSC BRNN in our future work.

Table 6.2 shows the results in detail.

6.3.2 Combining CNN and RNN

Finally, we combine existing ER-CNN [**Vu2016**] and our C-BRNN model. Our intuition is that their different way of sentence processing (convolution and max pooling vs. word-by-word processing and recurrence of hidden layer) can lead to performance gains through combination. The combination is performed with a voting process, as discussed in section 4.4. For each sentence in the test set, we apply existing ER-CNNs [**Vu2016**] with F1 score 84.2 and our C-BRNN model, presented in Table 6.2 and take that class which gets the most votes by them. In case of a tie, we pick one of the most frequent classes randomly. The combination achieves an F1 score of 84.9. This result is better than the performance of the two neural network types alone. It, thus, confirms our assumption that the networks provide complementary information. Further, in Fig 6.1, the performance of these two networks can be demonstrated based on the context length.

6.3.3 Comparison to state-of-the-art performance

State-of-the-art results for relation classification on SemEval data set		
Classifier	Features	F1
SVM ([**Hendrickx2009**])	POS, prefixes, morphological, WordNet, dependency parse, Levin classes, Probank, FrameNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
RNN	word embs + linguistic features	77.6
MVRNN	word embs ([**Collobert2008**]) + syntactic parse tree	79.1
([**Socher2012**])	+ POS + NER + WordNet	82.4
CNN	word embs([**Turian2010**])	69.7
([**Zeng2014**])	+position embs	78.9
	+WordNet	82.7
CNN ([**Nguyen2015**])	word embs([**Mikolov2013**]) + multiple window size + position emb	82.8
FCM	word embs	80.6
([**Yu2014**])	+ dependency parse tree + NER	83.0
bi-RNN+pool	word embs ([**Turian2010**]) (dim=50) + PI	80.0
([**Zhang2015**])	word embs ([**Mikolov2013**]) (dim=300) + PI	82.5
CR-CNN ([**Santos2015**])	word embs + position emb	84.1
ER-CNN ([**Vu2016**])	word embs + position emb	**84.2**
R-RNN	word embs + position indicators	**82.5**
ER-CNN+R-RNN	(see above)	**84.9**

Table 6.3: State-of-the-art results for relation classification on SemEval data set

Table 6.3 shows the results of the existing Extended Ranking-CNN (ER-CNN) model [**Vu2016**] from CIS LMU and our Ranking-RNN (R-RNN). Our proposed R-RNN (C-BRNN + ranking) is comparable to bi-RNN+pooling [**Zhang2015**], while it's combination with ER-CNN without using any linguistic features leads to the state-of-art performance as compared to other models, such as SVM [**Rink2010**], RNN and MVRNN [**Socher2012**], CNN [**Zeng2014**], FCM [**Gormley2014**], bi-RNN [**Zhang2015**] and CR-CNN [**Santos2015**].

6.3.4 Hyperparameters and training

The Table 6.4 Model parameter settings for C-BRNN shows the optimal model parameters settings, which we used for the top performing RNN model.

Hyperparamters	**Values**
RNN_model	*C-BRNN*
Ranking	*True*
weight sharing strategy	*independent*
Shuffle	*True*
# hidden units	*1000*
# hidden layer	*1*
# training epoch	*50*
initial learning rate	*0.01*
L2 weight	*0.0001*
mini batch size	*1*
optimizer	*gradient descent*
gradient norm clipstyle	*rescale*
gradient norm cutoff	*10.0*
position indicators	*True*
position embedding	*False*
entity flags	*False*
Extended context (Left neighborhood)	*5*
Extended context (Right neighborhood)	*5*
word embedding dimension (emb dim)	*400*
N-grams	*5*
weight initialization	*identity*
activation	*cappedRelu*

Table 6.4: Optimal Model parameter settings for C-BRNN

6.4 Analysis on RNN and CNN performance

6.4.1 Sentence lengths: Impact of context lengths

As demonstrated in section 5.6.1 for slot filling dataset, RNN has advantage over CNN model to capture the semantic information from large contexts. Specially, C-BRNN deals with long distance patterns more effectively by recursively learning word-by-word and propagating contribution of distance patterns to the last hidden unit, h_{bi} in Fig 4.10, where the relation label is attached.

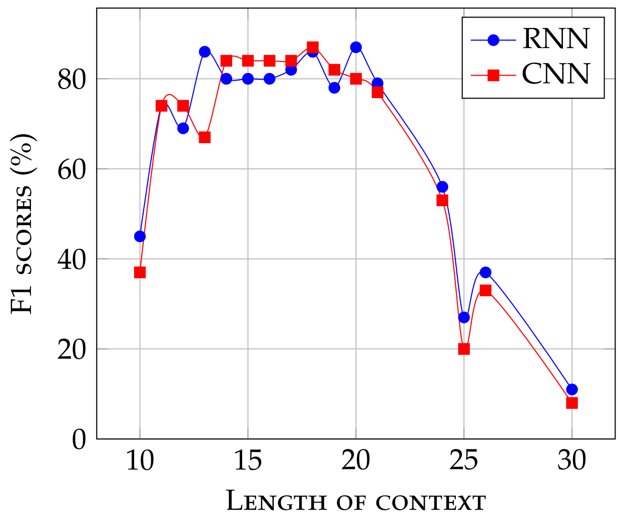

Figure 6.1: F1 scores of RNN and CNN with different context lengths.

We perform an analysis of RNN and CNN performance based on different sentence lengths, similar to section 5.6.1. The sentences in the test dataset of SemEval10 Task 8 are indexed by the context lengths. Then, we split the test dataset into the number of various test sentence lengths. As mentioned in section 5.6.1, the context includes the words between the two nominals and the extended context, i.e. 5 words prior to the first nominal and 5 words after the second nominal, if they exist. The context length also includes 4 additional the position indicators for the relation arguments in the sentence. Both models have the same group of samples to compute the F1 scores.

Similar to Fig 5.6.1, it can be seen in Fig 6.1 that if the context length is small, the CNN and RNN models perform similar, whereas for the larger context lengths, the RNN outperforms CNN. This confirms that RNN is more suitable to learn long-distance patterns. Note that with both the two models, the best F1 results are obtained with

Dataset	Context Length	Number of sentences
SemEval10 Task 8 (Testset)	< 10	56
	11-15	945
	16-20	1448
	> 20	268

Table 6.5: The distribution of context length in SemEval10 Task 8 testing dataset.

a moderate length of contexts. This is understandable as too small context involves limited semantic information, while too large context leads to difficulties in pattern learning. The distribution of the context length in SemEval10 Task 8 testing dataset is mentioned in the Table 6.5

6.4.2 Semantic accumulation by C-BRNNs

RNNs are temporal models that accumulate the semantic meaning of a sentence by learning recursively word-by-word at each time point. While, CNN model learns only local patterns.

Here, we demonstrate the nature of recurrent neural networks, using C-BRNN architecture, which captures the semantic meaning of sentence word-by-word. The peaks in the figures (Fig 6.3, 6.2, 6.4, 6.5, 6.6, 6.7) indicate the importance of a word or word segment (pattern) in representing the semantics of the sentence. For instance,

- A sentence from SemEval10 dataset:

  The current Iraqi government is seeking financial compensation from Israel for the <e1>damage</e1> caused by the <e2>bombing</e2> of the Osirak nuclear reactor in 1981.

  Since, we take extended context using the neighboring words to entity mentions for training, development and evaluation sets, i.e. 5 words left to <e1> and 5 words to the right of </e2>. Also, position indicators are treated as independent words in the word sequence. Therefore, word sequence

  $S2$: *compensation from Israel for the <e1> damage </e1> caused by the <e2> bombing </e2> of the Osirak nuclear reactor*

  We observe that the RNN captures the semantics word-by-word and peaks, in Fig 6.2, are observed at word inputs '</e1>', 'caused', 'by', and 'the'. The RNN has cumulative nature and accumulates the contribution of history words at each time step. Therefore, peak at the input word 'the' has contribution of the

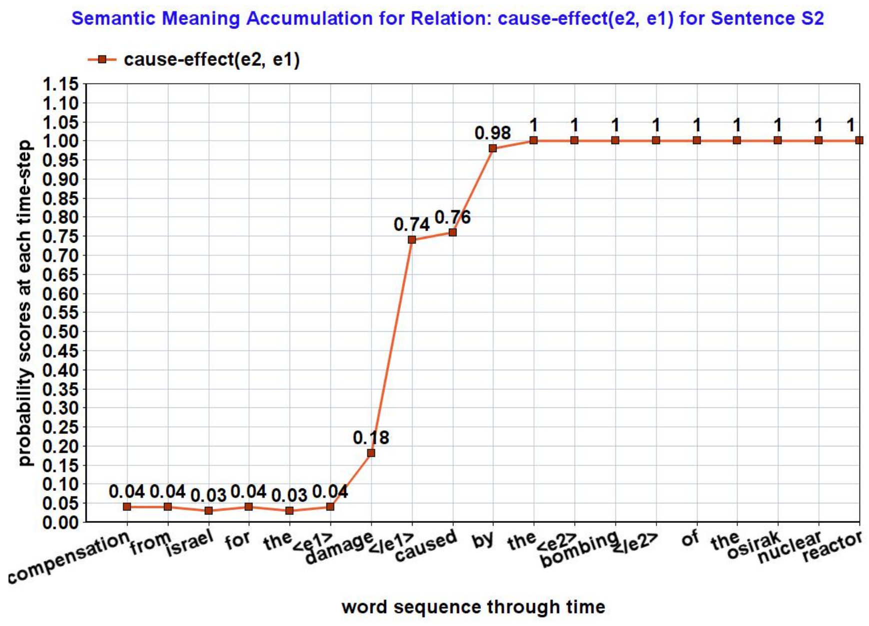

Figure 6.2: Semantic Meaning Accumulation by C-BRNNs for *cause-effect(e2, e1)*.

history word sequence .i.e. ’<e1> caused by’. These peaks also represent the representative N-Gram patterns i.e. ’caused by the’ that can be detected and is shown in Table 6.7.

To compute the probabilities (on y-axis) in figures (Fig 6.3, 6.2, 6.4, 6.5, 6.6, 6.7), we take our C-BRNN trained model and perform prediction on the given input the word/word segments. For example, the Table 6.6 and Fig 6.3 demonstrate how we captured the semantic nature of recurrent neural networks for word sequence *S*1 : *<e1> demolition </e1> was the cause of <e2> terror </e2>* for relation *cause-effect(e1, e2)*.

We found the analysis interesting and it also contributes to one of the novelty components of our thesis work.

6.4.3 Representative N-grams detected

In Tables (6.7, 6.8, 6.9, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15), we present an interesting analysis on the SemEval10 Task 8 dataset and list the top 5 N-Grams (3-gram, 5-gram and 7-gram) detected for each relation type, from the best performing C-BRNN model. These patterns contributed the most for scoring correctly classified examples and peaks in Figures (6.3, 6.2, 6.4, 6.5, 6.6, 6.7) indicate the importance of these patterns in the

Input word sequence to C-BRNN	prediction probability
<e1>	0.1
<e1>, **demolition**	0.25
<e1> demolition **</e1>**	0.29
<e1> demolition </e1> **was**	0.3
<e1> demolition </e1> was **the**	0.35
<e1> demolition </e1> was the **cause**	0.39
<e1> demolition </e1> was the cause **of**	0.77
<e1> demolition </e1> was the cause of **<e2>**	0.98
<e1> demolition </e1> was the cause of <e2> **terror**	1.0
<e1> demolition </e1> was the cause of <e2> terror **</e2>**	1.0

Table 6.6: Semantic Meaning Accumulation by C-BRNN word-by-word for *cause-effect(e1, e2)*. Bold indicates the last word in word segment input to C-BRNN.

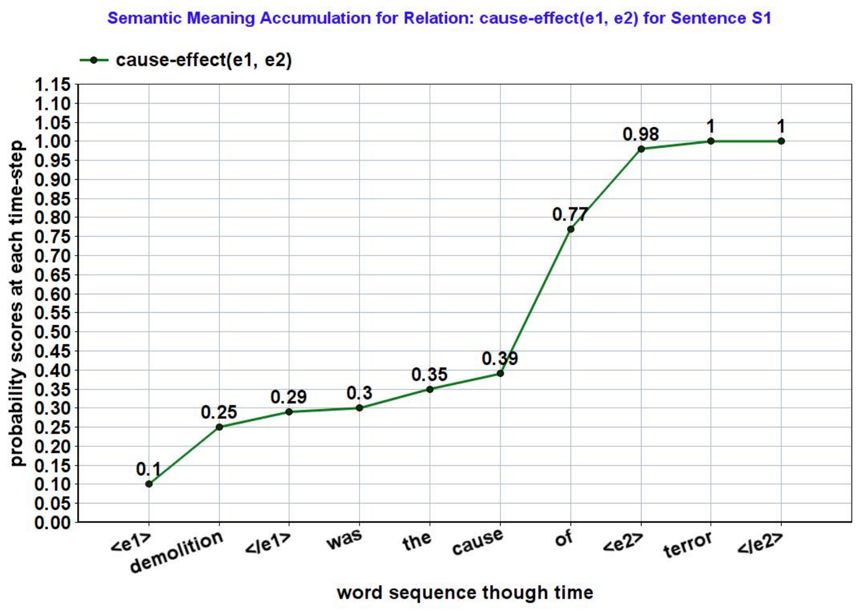

Figure 6.3: Semantic Meaning Accumulation by C-BRNNs for *cause-effect(e1, e2)*.

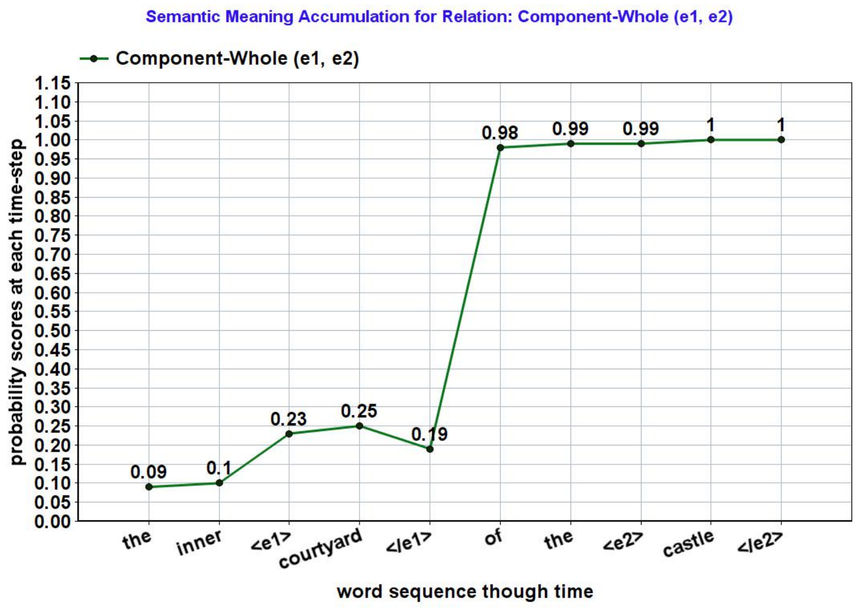

Figure 6.4: Semantic Meaning Accumulation by C-BRNNs for *component-whole(e1, e2)*.

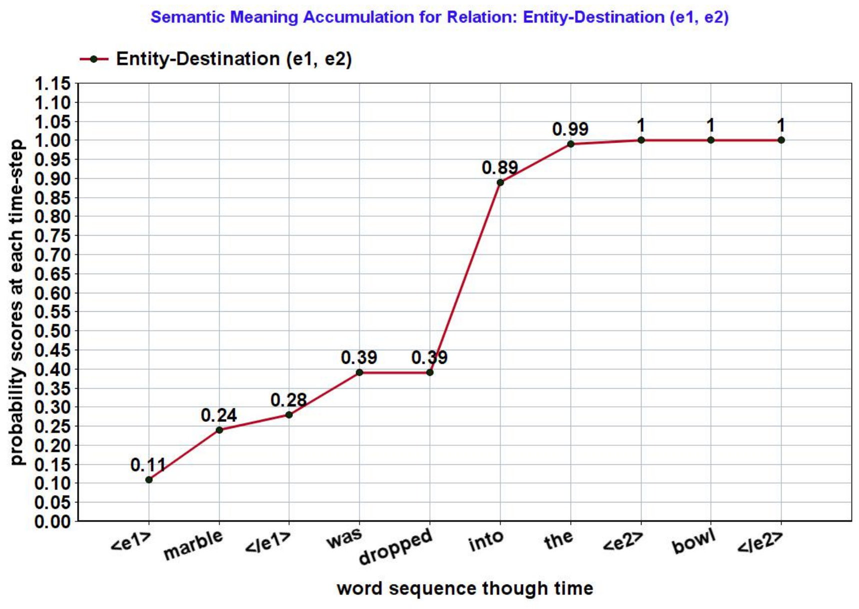

Figure 6.5: Semantic Meaning Accumulation by C-BRNNs for *entity-destination(e1, e2)*.

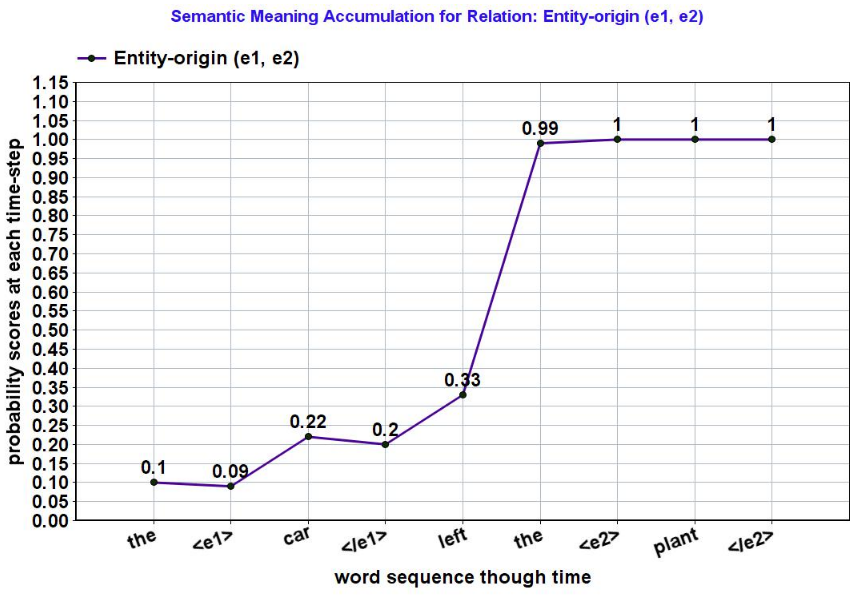

Figure 6.6: Semantic Meaning Accumulation by C-BRNNs for *entity-origin(e1, e2)*.

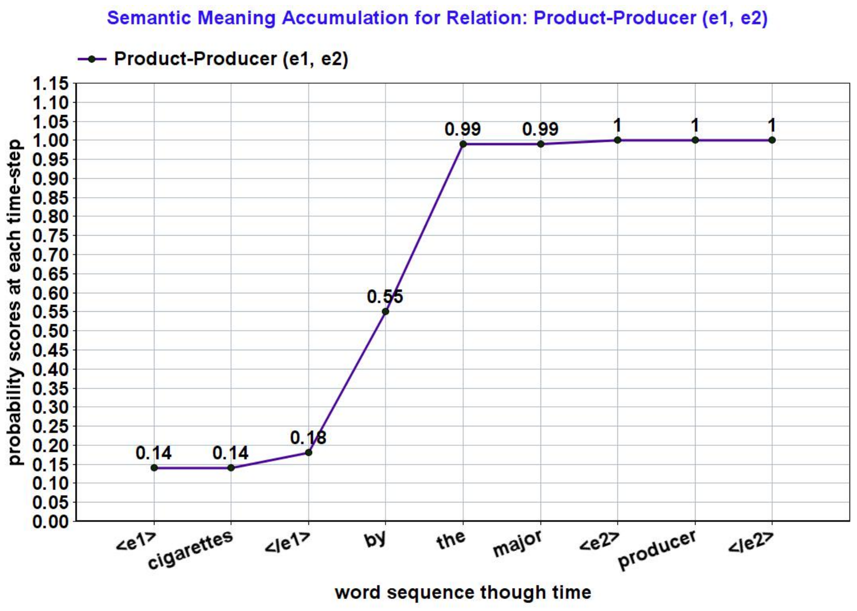

Figure 6.7: Semantic Meaning Accumulation by C-BRNNs for *product-producer(e1, e2)*.

sentence classification task. The most representative N-gram in a sentence is the one with the largest contribution to the improvement of the score.

To compute these N-grams, we provide an input word segment generated from sliding a window of N-gram through the sentence .i.e. a sliding window of size 3 words for trigrams, 5 for 5-gram and 7 for 7-gram patterns. The probability scores are computed from each word segment and the most representative N-grams are extracted, which contributes the most to the correct relation prediction. We have generated a number of patterns for each relation type and these patterns could be used as seed for relation extraction tasks.

While extracting N-gram patterns, we used position indicators as independent words in the sentences. One can observe that some relation types contain long representative patterns, for instance, in Table 6.10 for relation type *Entity-Origin (e1, e2)*, the 7-gram pattern ("*</e1> have been moving into <e2>*") and 5-gram pattern ("*</e1> moved into the <e2>*"), captures more informative longer patterns as compared to 3-gram patterns ("*</e2> into the*"). The position indicators are defined in section 3.2.4.

As expected, different N-grams play an important role depending on the direction of the relation. For instance, in table 6.11, the most informative trigram for *Entity-Origin(e1,e2)* is "*derived from an*", while reverse direction of the relation *Entity-Origin(e2,e1)* or *Origin-Entity*, has "*the source of*" as the most representative trigram. In addition, 5-gram captures "*is going away from the*" and "*which is the basic ingredient used for*" as the most informative longer patterns.

6.4.4 T-SNE visualizations

Here, we visualize the last hidden unit (h_{bi_N} in Fig 4.12) using t-SNE [**Maaten2008**]. For each sentence with a relation label in the training and test sets, we compute the last hidden unit (h_{bi_N}) that represents the semantics of the sentence accumulated by C-BRNN. Fig 6.8 represents semantic accumulated for relations (10 classes) without directions, while Fig 6.9 for relations (19 classes) with directions, each for SemEval10 task 8 training and testing datasets. Each color represents a relation-type and distinctive clusters are observed. Thus, the C-BRNN model has learned discriminative features for each relation.

Relations	N-Grams	N-Gram Patterns Detected
Cause-effect(e1,e2)	3-Grams	[</e1> cause <e2>] [</e1> caused a] [that cause respiratory] [which cause acne] [leading causes of]
	5-Grams	[the leading causes of <e2>] [the main causes of <e2>] [</e1> leads to <e2> inspiration] [</e1> that results in <e2>] [</e1> resulted in the <e2>]
	7-Grams	[is one of the leading causes of] [is one of the main causes of] [</e1> that results in <e2> hardening </e2>] [</e1> resulted in the <e2> loss </e2>] [<e1> sadness </e1> leads to <e2> inspiration] [</e1> is the source of an <e2>]
Cause-effect(e2,e1)	3-Grams	[caused due to] [comes from the] [arose from an] [caused by the] [radiated from a]
	5-Grams	[</e1> has been caused by] [</e1> are caused by the] [</e1> arose from an <e2>] [</e1> caused due to <e2>] [infection </e2> results in an]
	7-Grams	[</e1> is caused by a <e2> comet] [</e1> however has been caused by the] [</e1> that has been caused by the] [that has been caused by the <e2>] [<e1> product </e1> arose from an <e2>]

Table 6.7: Top 5 representative N-Grams (3, 5, 7) detected for *cause-effect* relation in SemEval10 dataset by C-BRNNs with PI

Relations	N-Grams	N-Gram Patterns Detected
Component-Whole(e1,e2)	3-Grams	[</e1> of the]
		[of the <e2>]
		[part of the]
		[</e1> of <e2>]
		[</e1> on a]
	5-Grams	[</e1> of the <e2> device]
		[</e1> was a part of]
		[</e1> is part of the]
		[is a basic element of]
		[is part of a]
	7-Grams	[the <e1> timer </e1> of the <e2>]
		[</e1> was a part of the romulan]
		[</e1> was the best part of the]
		[</e1> is a basic element of the]
		[are core components of the <unk> solutions]
Component-Whole(e2,e1)	3-Grams	[with its <e2>]
		[′ s <e2>]
		[contains a <e2>]
		[comprises the <e2>]
		[</e1> has a]
	5-Grams	[</e1> ′ s <e2> pouch]
		[</e1> briefly comprises of <e2>]
		[</e1> has a large <e2>]
		[</e1> was made with <e2>]
		[</e1> includes a 0 -]
	7-Grams	[</e1> briefly comprises of <e2> hall </e2>]
		[</e1> is made with gelatinous veal <e2>]
		[<e1> fish </e1> with <e2> lungs </e2>]
		[</e1> has a <e2> coil </e2> with]
		[</e1> "" s <e2> voice </e2> signals]

Table 6.8: Top 5 representative N-Grams (3, 5, 7) detected for *Component-Whole* relation in SemEval10 dataset by C-BRNNs with PI

Relations	N-Grams	N-Gram Patterns Detected
Content-Container(e1,e2)	3-Grams	[in a <e2>] [was inside a] [contained in a] [hidden in a] [stored in a]
	5-Grams	[</e1> was contained in a] [</e1> was discovered inside a] [</e1> were in a <e2>] [is hidden in a <e2>] [</e1> was contained in a]
	7-Grams	[</e1> was contained in a <e2> box] [</e1> was in a <e2> suitcase </e2>] [</e1> were in a <e2> box </e2>] [</e1> was inside a <e2> box </e2>] [</e1> was hidden in an <e2> envelope]
Content-Container(e2,e1)	3-Grams	[the <e1> envelope] [</e1> with <e2>] [full of <e2>] [the <e1> pouch] [</e1> contained a]
	5-Grams	[</e1> full of dog <e2>] [<e1> bottle </e1> full of] [was full of fine <e2>] [box </e1> contained a dollar] [bag </e1> with <e2> condoms]
	7-Grams	[with a <e1> bottle </e1> full of] [</e1> with <e2> condoms </e2> in it] [the <e1> pouch </e1> contained a small] [is a <e1> bottle </e1> with <e2>] [was a <e1> bottle </e1> with <e2>]

Table 6.9: Top 5 representative N-Grams (3, 5, 7) detected for *Content-Container* relation in SemEval10 dataset by C-BRNNs with PI

Relations	N-Grams	N-Gram Patterns Detected
Entity-Destination(e1,e2)	3-Grams	[put into a] [released into the] [</e1> into the] [moved into the] [added to the]
	5-Grams	[have been moving into the] [was dropped into the <e2>] [</e1> moved into the <e2>] [were released into the <e2>] [</e1> have been exported to]
	7-Grams	[</e1> have been moving back into <e2>] [</e1> have been moving into the <e2>] [</e1> have been dropped into the <e2>] [</e1> have been released back into the] [power </e1> is exported to the <e2>]
Entity-Destination(e2,e1)	3-Grams	-
	5-Grams	-
	7-Grams	-

Table 6.10: Top 5 representative N-Grams (3, 5, 7) detected for *Entity-Destination* relation in SemEval10 dataset by C-BRNNs with PI

Relations	N-Grams	N-Gram Patterns Detected
Entity-Origin(e1,e2)	3-Grams	[derived from an] [away from the] [left from the] [arrived from the] [ran away from] [off from an]
	5-Grams	[</e1> is derived from a] [is going away from the] [</e1> had left the <e2>] [was delivered from the <e2>] [</e1> arrived from the <e2>]
	7-Grams	[</e1> is derived from a static <e2>] [</e1> is going away from the display] [<e1> student </e1> was released from <e2>] [</e1> arrived from an <e2> executive </e2>] [train </e1> departed from <e2> station </e2>]
Entity-Origin(e2,e1)	3-Grams	[also had some] [have also had] [the source of] [source of an] [<e1> grape </e1>]
	5-Grams	[<e1> chestnut </e1> <e2> flour] [</e1> which is the basic] [</e1> is the source of] [the source of the most] [, which appears to have]
	7-Grams	[which is the basic ingredient used for] [is the source of the most familiar] [</e1> is the source of the <e2>] [soup </e2> is one of the best] [<e1> grape </e1> <e2> wine </e2> ,]

Table 6.11: Top 5 representative N-Grams (3, 5, 7) detected for *Entity-Origin* relation in SemEval10 dataset by C-BRNNs with PI

Relations	N-Grams	N-Gram Patterns Detected
Instrument-Agency(e1,e2)	3-Grams	[</e1> are used] [used by <e2>] [</e1> is used] [set by the] [</e1> set by]
	5-Grams	[</e1> assists the <e2> eye] [</e1> are used by <e2>] [</e1> were used by some] [</e1> with which the <e2>] [readily associated with the <e2>]
	7-Grams	[cigarettes </e1> are used by <e2> women] [<e1> telescope </e1> assists the <e2> eye] [<e1> practices </e1> for <e2> engineers </e2>] [the best <e1> tools </e1> for <e2>] [<e1> wire </e1> with which the <e2>]
Instrument-Agency(e2,e1)	3-Grams	[</e1> used <e2>] [with a <e2>] [attempted by <e1>] [by a <e1>] [by using <e2>]
	5-Grams	[by using <e2> snowshoes </e2>] [washer </e1> works with a] [killed her with a <e2>] [with a <e2> candle </e2>] [</e1> who tried to steal]
	7-Grams	[</e1> work by using <e2> electricity </e2>] [with the help of the <unk> <e2>] [without causing excessive damage by using <e2>] [with a number of <e2> <unk> </e2>] [</e1> killed her with a <e2> sword]

Table 6.12: Top 5 representative N-Grams (3, 5, 7) detected for *Instrument-Agency* relation in SemEval10 dataset by C-BRNNs with PI

Relations	N-Grams	N-Gram Patterns Detected
Member-Collection(e1,e2)	3-Grams	[group </e2> of]
		[of the <e2>]
		[</e1> in the]
		[</e1> of this]
		[collected in this]
	5-Grams	[part of their <e2> association]
		[head </e1> of the <e2>]
		[leader </e1> of this <e2>]
		[</e1> of this <e2> organization]
		[</e1> collected in this <e2>]
	7-Grams	[</e1> is a member of the <e2>]
		[</e1> was a member of the <e2>]
		[</e1> collected in this <e2> volume </e2>]
		[leader </e1> of that <e2> organization </e2>]
		[</e1> in his <e2> party </e2> .]
Member-Collection(e2,e1)	3-Grams	[</e1> of <e2>]
		[army </e1> of]
		[filled with a]
		[</e1> consists of]
		[was composed of]
	5-Grams	[a <e1> battalion </e1> of]
		[ensemble </e1> of <e2> ladies]
		[</e1> consists of edited <e2>]
		[a member of the <e1>]
		[taken in by a <e1>]
	7-Grams	[consists of an <e1> ensemble </e1> of]
		[</e1> of <e2> musicians </e2> artists]
		[<e1> panel </e1> <e2> member </e2> on]
		[joining the <e1> collective </e1> of]
		[filled with a <e1> kindle </e1> of]
		[</e1> was composed of elected <e2> representatives]

Table 6.13: Top 5 representative N-Grams (3, 5, 7) detected for *Member-Collection* relation in SemEval10 dataset by C-BRNNs with PI

Relations	N-Grams	N-Gram Patterns Detected
Message-Topic(e1,e2)	3-Grams	[</e1> inform about] [<e1> outline </e1>] [</e1> define <e2>] [</e1> is the] [</e1> asserts about]
	5-Grams	[</e1> to inform about <e2>] [</e1> defines <e2> purpose </e2>] [</e1> outlines the <e2> duties] [article </e1> gives details on] [<e1> slides </e1> explaining <e2>]
	7-Grams	[</e1> to remind them about the <e2>] [<e1> documents </e1> define <e2> policies </e2>] [<e1> book </e1> asserts the <e2> notion] [the <e1> report </e1> informed about <e2>] [presentation </e1> is the <e2> status </e2>]
Member-Collection(e2,e1)	3-Grams	[been clearly explained] [discussed on message] [details </e1> of] [have been described] [</e1> being discussed]
	5-Grams	[has been reflected in <e2>] [</e1> has been narrated in] [was analysed in an <e2>] [described in the <e2> literature] [being discussed at the <e2>]
	7-Grams	[was reflected in <e2> comments </e2> published] [<e1> information </e1> is found in the] [have been discussed in the <e2> section] [has been documented in a recent <e2>] [has been featured in numerous <e2> publications]

Table 6.14: Top 5 representative N-Grams (3, 5, 7) detected for *Message-Topic* relation in SemEval10 dataset by C-BRNNs with PI

Relations	N-Grams	N-Gram Patterns Detected
Product-Producer(e1,e2)	3-Grams	[</e1> released by] [</e1> issued by] [</e1> created by] [by the <e2>] [of the <e1>]
	5-Grams	[</e1> issued by the <e2>] [</e1> was prepared by <e2>] [was written by a <e2>] [of the <e1> products </e1>] [</e1> built by the <e2>] [</e1> are made by <e2>]
	7-Grams	[<e1> products </e1> created by an <e2>] [</e1> by an <e2> artist </e2> who] [</e1> written by most of the <e2>] [temple </e1> has been built by <e2>] [</e1> were founded by the <e2> potter]
Product-Producer(e2,e1)	3-Grams	[′ s <e2>] [company </e1> manufactured] [<e2> of the] [makes an <e2>] [</e1> came up]
	5-Grams	[</e1> ′ s products have] [</e1> has made a <e2>] [</e1> were producing <e2> products] [</e1> who did the <e2>] [</e1> also came up with]
	7-Grams	[the <e1> factory </e1> ′ s products] [</e1> came from the <e2> chairman </e2>] [the <e1> corporation </e1> has constructed <e2>] [<e1> manufacturer </e1> ′ s <e2> drug] [</e1> synthesizes a <e2> copy </e2> of]

Table 6.15: Top 5 representative N-Grams (3, 5, 7) detected for *product-producer* relation in SemEval10 dataset by C-BRNNs with PI

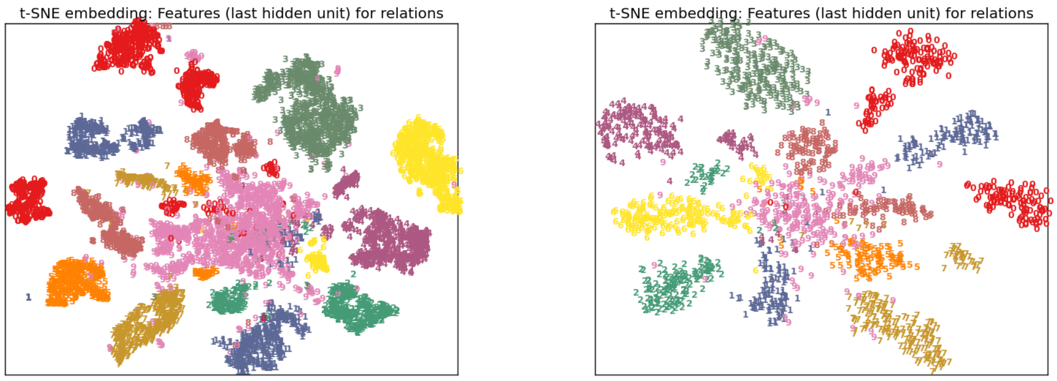

Figure 6.8: t-SNE Visualization of the last combined hidden unit (h_{bi}) of BRNN for SemEval10 Training (Left) and Testing (Right) dataset for 10 relations without directions. Each color has at least two clusters, representing the same relation with no direction. For instance, *red* numbered 0 or *pale yellow* numbered 7 in (Right). *Magenta* cluster in the center represents the *Other* class.

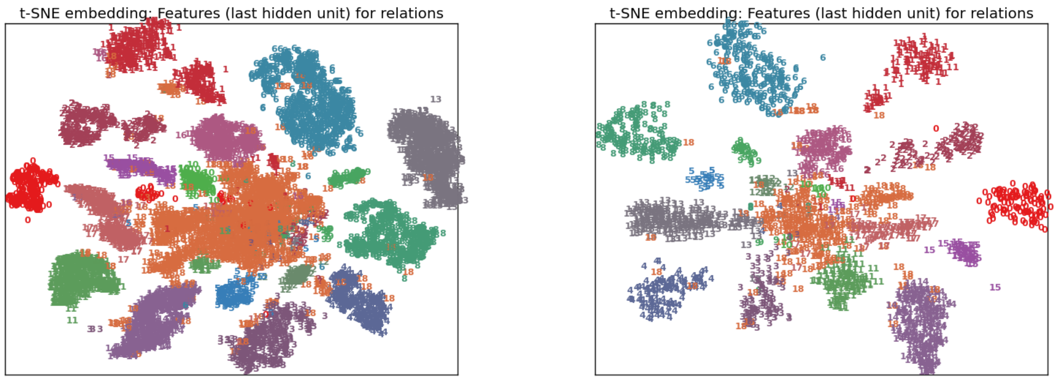

Figure 6.9: t-SNE Visualization of the last combined hidden unit (h_{bi}) of BRNN for SemEval10 Training (Left) and Testing (Right) dataset for 19 relations with directions. Every color has mostly a single cluster representing unique direction. For instance, *red* numbered 0 and *wine red* numbered 1 in (Right). *Dark brown* cluster in the center represents the *Other* class.

7 Outcomes of this thesis

7.1 Technical contributions

- Novel RNN architectures: Connectionist bi-directional RNN (C-BRNNs) and Connectionist Multi-step-context RNN (C-MSC BRNNs) for relation classification, leading to improved performance compared to standard bi-directional RNNs.
- Introduced ranking in C-BRNNs (R-RNNs) [**Vu2016**]
- Joint forward and backward networks training in C-BRNN.
- Investigation on independent, shared and transposed weights in forward and backward networks of BRNN for relation classification.
- Introduction of N-gram input to RNNs, resulting in superior results for relation classification, as compared to single word vector input.
- Most representative N-gram patterns detection.
- Demonstrated the semantic meaning accumulation nature of RNNs.
- Demonstrated the semantic composition-accumulation-propagation of patterns in C-BRNNs

7.2 Experiments on SemEval10 dataset

- Achieved the state-of-art performance from the proposed C-BRNN+ranking model combined with existing CNN [**Vu2016**]. Results submitted in AAAI 2016 [**Vu2016**].
- Most representative N-gram patterns detection and extraction for all 19 relations using C-BRNN model.
- Analysis of CNN vs RNN performance based on the different sentence lengths

7.3 Experiments on Slot Filling dataset

- Introduced RNNs into slot filling task
- Integration of RNNs in slot filling pipeline ([**Adel2014**])
- A number of experiments (individual model for each slot i.e. 22 slots) run for SF 2014 as well 2015 evaluations.
- Improved results for some slots in SF dataset using the shared and transposed weight sharing strategies.
- Most representative N-gram patterns detection and extraction for slots *per:spouse* and *per:location_of_birth* using C-BRNN model.
- Analysis of CNN vs RNN performance based on the different sentence lengths
- Improved scores (*preliminary*) for the 2015 TAC KBP SF submission, from RNNs in combination with existing PAT+SVM+CNN ([**Adel2014**]).

8 Summary

In this thesis work, we investigated standard recurrent neural networks for relation classification and introduced a bi-directional recurrent neural network, named as Connectionist BRNN (C-BRNN) in Fig 4.10. We showed that the recurrent learning models can solve the natural language tasks involving sentence-level predictions without using any linguistic features from the existing NLP tools. The automatic feature learning approach by RNNs help to generalize through various application domains and deal with the difficulties in the pattern design, the lack of annotated data and high quality feature engineering through NLP tools. The proposed model is investigated for relation classification task on the SemEval10 and TAC KBP SF datasets.

The chapter 3 described the model training and key features investigated for relation classification task. Section 3.1.2 explained the ranking function, which is introduced in the output layer of C-BRNN model, leading to superior results as compared to softmax output layer. Initial investigations with uni-directional RNNs are performed on features such as word representation, position features (position embeddings and position indicator embeddings), word features (i.e. N-gram input word vectors) and extended context, as discussed in section 4.1. From these investigations, it is observed that position indicators along with word features improve the scores for relation classification task on SemEval10 dataset. We performed experiments with the possible choices of activation functions and model initialization strategies discussed in section 3.3, which handled the vanishing and exploding gradient problems during RNN optimization.

Chapter 4 turned to the designs of recurrent neural network for relation classification task. Following the uni-directional RNNs, we further investigated existing bi-directional RNNs and proposed C-BRNN model for relation classification task, which leads to significant improvements. The connectionist network introduced time-step feedback connections among the combined hidden units, which allowed the accumulated semantics from forward and backward network to propagate to the end of the network (Fig 4.10). These C-BRNNs are further investigated using position indicators and word features along with ranking, providing special treatment to *Other* class in SemEval10 Task 8 dataset. In chapter 4, we also discussed the need of bi-directional networks for relation classification and demonstrated the error backpropagation through time in C-BRNN (Fig 4.12), reasoning the relevance of all the combined hidden units from forward and backward networks, as compared to only the last combined hidden unit in standard BRNN (Fig 4.5) in the context of relation classification. Here, we also mentioned the joint forward and backward network training in C-BRNNs. In particular, the chapter also talked about the treatment of weight matrices in the forward and backward networks of bi-directional RNNs. We found the performance of independent, shared and transposed weights in the forward and backward networks is comparable on SemEval10 dataset, detailed in Table 6.2. CNN and RNN have different learning approach, therefore, in order to advantage from both the models, the combination of RNN and existing CNN by voting and weighted combination is described in section 4.4.

The introduction of RNN into slot filling task is discussed in chapter 5. We investigated the performance of uni-directional and bi-directional RNNs for the TAC KBP SF 2014 and 2015 evaluation queries. RNN is integrated in the slot filling pipeline (developed by CIS LMU) and for each slot, we report F1 scores of combining the RNN with the existing pattern-based approach (PAT), SVM and CNN. For the 2014 TAC KBP SF evaluation, the combination PAT+SVM+CNN+RNN considerably improves the individual classification results for most of the slots (Tables 5.4 and 5.5), while the overall performance gain in the slot filling run is minimal. As observed in Table 5.6, RNNs improve the recall. In addition, we also participated in the 2015 TAC KBP SF research task with CIS LMU team. The published scores (preliminary), in Table 5.7, shows that the combination PAT+SVM+CNN+RNN slightly performs superior to PAT+SVM+CNN.

The chapter 5 and chapter 6 discussed about the interesting analysis performed on SF and SemEval10 datasets, respectively. In sections 5.6.1 and 6.4.1, we analyzed the impact of different context lengths on the performance of the CNN and RNN models for SF and SemEval10 datasets. Fig 5.2 and Fig 6.1 showed that the RNN is superior to CNN in performance for the large sentence lengths. We also demonstrated the accumulating behavior of RNNs, where they process sentences word-by-word at each time-step and accumulate the semantic meaning, as in Fig 5.4 and Fig 6.3. Following these analysis, we also detected the most representative tri-grams, 5-grams and 7-grams that significantly contributes to the correct relation prediction. Table 6.7 pointed that RNN has the capability to capture the distant long term patterns i.e. 5-grams and 7-grams. The analysis performed is also one of the major contributions of this thesis work.

The Table 6.2 summarized various experimental results for relation classification on SemEval10 task. The combination of the proposed C-BRNN+ranking (R-RNN) and the existing ER-CNN models result in the state-of-art performance (Table 6.3). The scores are further submitted to AAAI-16 conference ([**Vu2016**]).

9 Future work

In our thesis work, we investigated RNN models without using linguistic features from existing NLP tools, such as part-of-speech (POS), named-entity recognizer (NER), syntactic parse trees and WordNet. In future, we would like to investigate the performance of the proposed Connectionist bi-directional RNN (C-BRNN) including these linguistic features for relation classification task. In our investigations, the C-BRNN model has a single layer of recurrent hidden units, while it is worth investigating the deep recurrent neural networks, by stacking the layers on the top of each other in depth, as evaluated for opinion mining ([**Ozan2014**]) task. Using SemEval10 data, we made initial investigations on multi-step RNNs to capture the long term dependencies. However, the performance of C-MSC BRNN models with LSTM units for relation classification task could be explored.

The combination of existing CNN and proposed RNN leads to the state-of-art performance for relation classification on SemEval10 Task 8 dataset. Here, each model is trained independently and the decisions are combined using voting strategy for SemEval10 dataset, while weighted combination for the slot filling task. Therefore, in order to leverage the advantages of the two types of learning in a combined network, the joint model training of CNN and RNN is open to investigate.

References

[Adel2014] Heike Adel and Hinrich Schütze: TAC KBP 2014 Slot Filling Shared Task: Baseline System for Investigating Coreference, TAC 2014

[Adel2016] . Heike Adel and Hinrich Schütze. 2015. A Comparison of Three Approaches to Slot Filling: Patterns, Bag-of-words and Convolution. Submitted to AAAI 2016.

[Agichtein2000] Agichtein and L. Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In Proceedings of the 5th ACM international Conference on Digital Libraries (ACMDL 2000).

[Alex2010] Alex Krizhevsky. 2010. Convolution Deep Belief Network on CIFAR-10. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.5826rep=rep-1type=pdf

[Angel2014] G. Angel, J. Tibshirani, J. Wu, and C. D. Manning. 2014. Combining distant and partial supervision for relation extraction. In Proc. The 2014 Conference on Empirical Methods on Natural Language Processing).

[Angeli2014] Gabor Angeli, Sonal Gupta, Melvin Jose, Christopher D. Manning, Christopher, Julie Tibshirani, Jean Y. Wu, Sen Wu, Ce Zhang. 2014. Stanford's 2014 Slot Filling Systems.

[AngeliandGupta2014] Gabor Angeli, Sonal Gupta, Melvin Jose, Christopher D. Manning, Christopher, Julie Tibshirani, Jean Y. Wu, Sen Wu, Ce Zhang. 2014. Stanford's Distantly Supervised Slot Filling Systems for KBP 2014.

[Bastien2012] . Bastien, Lamblin, Pascanu, Bergstra, Goodfellow, Bergeron,, Bouchard, and Bengio Y. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

[Bengio1993] Bengio, Y., Frasconi, P., and Simard, P. 1993. The problem of learning long-term dependencies in recurrent networks. pages 1183–1195, San Francisco. IEEE Press.

[Bengio2003] Yoshua Bengio, Rejean Ducharme and Pascal Vincent. 2003. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137-1155

[Bengio2009] Yoshua Bengio. 2009. Learning deep architectures for AI. Foundations and Trends Machine Learning, 2(1):1–127.

[Bergstra2010] . Bergstra, Breuleux, Bastien, Lamblin, Pascanu, Desjardins, Turian, Warde-Farley, and Bengio Y. 2010. Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for Scientific Computing Conference (SciPy).

[Brin1998] S. Brin. 1998. Extracting patterns and relations from the world wide web. In WebDB Workshop at 6th International Conference on Extending Database Technology (EDBT 98).

[Collobert2011] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from, scratch. Journal of Machine Learning Research, 12:2493–2537.

[Elman1990] J. Elman. 1990. "Finding structure in time". In Proceedings of in Cognitive Science, 14 (2).

[Glorot2010] . Xavier Glorot, Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In proceedings of International conference on artificial intelligence and statistics.

[Gupta2014] Sonal Gupta and Christopher D. Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In Proceedings of the Eighteenth Conference on Computational Natural Language Learning (CoNLL).

[Harris1954] Harris, Zellig. 1954. Distributional structure. Word 10(23). 146–162.

[Hendrickx2010] Hendrickx, Kim, Kozareva, Nakov, , Sebastian, Pennacchiotti, Romano, and Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 33–38.

[Hu2014] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In Proceedings of the Conference on Neural Information Processing Systems, pages 2042–2050.

[Jacobs1995] R. A. Jacobs. 1995. Methods for combining experts' probability assessments. Neural Computing, vol. 7, no. 5, pp. 867–888.

[Jordan1986] M. Jordan. 1986. "Serial order: A parallel distributed processing approach". Published in Tech. Rep. No. 8604. San Diego: University of California, Institute for Cognitive Science.

[Kambhatla2014] Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions, page 22. Association for Computational Linguistics.

[Kim2014] Kim, Y. 2014. Convolutional neural netowrks for sentence classification. In Proceedings of EMNLP. Association for Computational Linguistics.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods for Natural Language Processing, pages 1746–1751, Doha, Qatar.

[Maaten2008] . L Van der Maaten, G Hinton. 2008. Visualizing data using t-SNE. In Proceedinggs of JMLR.

[Mesnil2013] Grégoire Mesnil, Xiaodong He, Li Deng and Yoshua Bengio. 2013. Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding. Interspeech.

[Mesnil2013] Grégoire Mesnil, Xiaodong He, Li Deng and Yoshua Bengio. Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding. Interspeech, 2013.

[Mihai2013] Mihai Surdeanu. 2013. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In Proceedings of the TAC-KBP 2013 Workshop.

[Mikolov2010] Mikolov, Karafiat, Cernocky, and Danpur. 2014. Recurrent neural network based language model. In Proceedings of Interspeech, 2010.

[Mikolov2013] Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. In Proceedings of the Workshop at ICLR.

[Miller2000] S. Miller, H. Fox, L. Ramshaw, and R. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In Proceedings of the 6th Applied Natural Language Processing Conference. 29 April-4 May 2000, Seattle, USA.

[Mooney2005] Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pages 724–731.

[Nguyen2015] Nguyen, T. H., and Grishman, R. 2015. Relation extraction: Perspective from convolutional neural networks. In Proceedings of the NAACL Workshop on Vector Space Modeling for NLP. Association for Computational Linguistics.

[Niu2012] Feng Niu, Ce Zhang, Christopher Re, and Jude W Shavlik. 2012. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. In Proceedings of VLDS.

[Ozan2014] Ozan Irsoy, C. Cardie. 2014. Opinion mining with Deep Recurrent Neural Networks. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 720–728.

[Paliwal1997] Schuster, M., and Paliwal, K. K. 1997. Bidirectional recurrent neural networks. Signal Processing, IEEE Transactions on.

[Pascanu2012] Pascanu, R.; Mikolov, T.; and Bengio, Y. 2012. Understanding the exploding gradient problem. Computing Research Repository.

[Pascanu2013] Razvan Pascanu, Yoshua Bengio. 2013. How to construct deep recurrent neural networks. arXiv preprint arXiv:1312.6026.

[Pershina2014] Maria Pershina, Bonan Min, Wei Xu, and Ralph Grishman. 2014. Infusion of labeled data into distant supervision for relation extraction. In Proceedings of ACL.

[Plank2013] Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 1498–1507

[Qian2008] L.H. Qian, G.D. Zhou, Q.M. Zhu, and P.D Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In Proceedings of The 22nd International Conference on Computational Linguistics (COLING 2008), pages 697-704. 18-22 August 2008, Manchester, UK.

[Qian2009] Kong, and Qiaoming Zhu. 2009. Semi-supervised learning for semantic relation classification using stratified sampling strategy. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 1437–1445.

[Quoc2015] Quoc V. Le, Navdeep Jaitly, Geoffrey E. Hinton. 2015. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. http://arxiv.org/pdf/1504.00941.pdf

[Rink2010] Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In Proceedings of International Workshop on Semantic Evaluation, pages 256–259.

[Roth2013] Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2013. Effective slot filling based on shallow distant supervision methods. In Proceedings of the Sixth Text Analysis Conference (TAC 2013).

[Roth2014] Benjamin Roth, Tassilo Barth, Grzegorz Chrupa, Martin Gropp, and Dietrich Klakow. 2014. Relationfactory: A fast, modular and effective system for knowledge base population. In Proceedings of Demonstrations at ACL.

[Santos2014] Santos and Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland.

[Santos2015] Dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In Proceedings of ACL-15, Beijing, China, 2015.

[SantosandZadrozny2014] Santos and Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In Proceedings of the 31st International Conference on Machine Learning (ICML), JMLR: WCP volume 32, Beijing, China.

[Schmidhuber1992] Jurgen Schmidhuber. 1992. Learning complex, extended sequences using the principle of history compression. Neural Computation, 4(2):234–242.

[Socher2012] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211.

[Surdeanu2012] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In Proceedings of the 2012 Conference on Empirical Methods in Natural Language

Processing and Natural Language Learning (EMNLP-CoNLL), pages 455– 465. ACL.

[Surdeanu2014] Mihai Surdeanu1, Heng Ji 2014. Overview of the English Slot Filling Track at the TAC2014 Knowledge Base Population Evaluation. In Proceedings of the TAC-KBP 2014 Workshop.

[TACKBP2015] http://www.nist.gov/tac/2015/KBP/ColdStart/index.html

[Tratz2010] Tratz, S., and Hovy, E. 2010. Isi: automatic classification of relations between nominals using a maximum entropy classifier. In Proceedings of the Workshop on SemEval. Association for Computational Linguistics.

[Vu2016] Ngoc Thang Vu, Heike Adel, Pankaj Gupta and Hinrich Schütze: Combining Recurrent and Convolution Neural Networks for Relation Classification, submitted to AAAI 2016.

[Wang2008] Mengqiu Wang. 2008. A re-examination of dependency path kernels for relation extraction. In Proceedings of the Third International Joint Conference on Natural Language Processing, pages 841–846.

[Werbos1990] Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. In Proceedings of the IEEE, 78(10):1550–1560

[word2Vec] code.google.com/p/word2vec/.

[Yarowsky1995] D. Yarowsky. Unsupervised word sence disambiguation rivaling supervised methods. In Meeting on the Association for the Computation linguatistics, pages 189-196, 1995

[Yu2014] Mo Yu, Matthew R. Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. In The NIPS 2014 Learning Semantics Workshop, 2014.

[Zelenko2003] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. The Journal of Machine Learning Research, 3:1083–1106.

[Zeng2014] Zeng, D., Liu, K., Lai, S., Zhou, G., and Zhao, J. 2014. Relation classification via convolutional deep neural network. In Proceedings of COLING.

[Zhang2004] Zhu Zhang. 2004. Weakly-supervised relation classification for information extraction. In Proceedings of the ACM International Conference on Information and Knowledge Management, pages 581–588, New York, NY, USA.

[Zhang2006] M. Zhang, J. Zhang, J. Su, and G.D. Zhou. 2006. A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association of Computational Linguistics (COLING/ACL 2006), pages 825-832. Sydney, Australia.

[ZhangandWang2015] Zhang, D., and Wang, D. 2015. Relation classification via recurrent neural network. In ArXiv.

[Zimmermann2001] Zimmermann H. G. and Neuneier R. 2001. Neural Network Architectures for the Modeling of Dynamical Systems, in: A Field Guide to Dynamical Recurrent Networks, Kolen, J. F.; Kremer, St. (Ed.); IEEE Press, 2001, pp. 311-350.

List of Figures

List of Tables