A Neural Approach for Thematic Role-Fillers Prediction Based on Context Information

Ruixue Liu

Supervisor: Prof. Dr. Vera Demberg
Dr. Asad Sayeed
Dr. Denis Paperno

University of Lorraine
Saarland University

This dissertation is submitted for the degree of
Master of Science in
Language Science and Technology

September 2017

Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Saarbrücken, Germany, _____ _____

Ruixue Liu
September 2017

Acknowledgements

First of all, I would like to thank all my supervisors, firstly Prof. Vera Demberg for her guidance and support for the project. Secondly, Asad Sayeed for his patience when helping explain many aspects of the research and his continued support despite moving to Sweden for a new position. I also appreciate all the valuable suggestions and help from Denis Paperno and his generosity for supervising me from University of Lorraine. With all their encouragement and guidance, my research skills have improved considerably.

Secondly, I would like to thank Xudong Hong, Quan Phan Ngoc, Stalin, Nima and Wei Shi for providing all the insightful comments, explanation and technical support during my project and helping steer it in the right direction. I also appreciate the suggestions and help from Fraser, Andrew and Katie Ann for improving my academic writing.

I would like to show my gratitude to Dr. Ben Amsel for allowing me access to the stimuli data of experiments and permission of modifying it for our purpose of evaluation.

My sincere thanks goes to the LCT coordinators, Jürgen Trouvain and Bobbye Pernice, for their help of making my study and life in Saarland University go smoothly. I also appreciate all the help from Maxime Amblard and Miguel Couceiro during my stay in University of Lorraine, France.

Finally, my biggest gratitude goes to my parents, my sister, Yuhang and all my friends, for their support, encouragement and love, without which, this thesis would not be possible.

Abstract

Thematic role-filler prediction refers to the prediction of certain words corresponding to given thematic roles, while selectional preference refers to a word's tendency to co-occur with other words that belong to certain lexical sets. Traditional approaches for thematic role-filler prediction or selectional preference rely either on hand-crafted resources such as word-net or on unsupervised machine learning mechanisms such as distributional similarity metrics. Throughout this research, we investigated the use of deep learning approaches for semantic role-filler prediction. Taking a large amount of automatically role-labeled text as input, the model is expected to predict a suitable role-filler given the target semantic role and the context of other role-fillers. In order to enable our model to present a high quality distributed representation of a semantic role under different contexts, the arguments of the predicate should take contextual information (e.g the entire noun phrase or sentence) into consideration. In our experiments, we explored various ways to embed important words contained in the noun phrase or sentence for semantic roles. The experimental results indicate that building factored rolespecific word embedding matrices and factored rolespecific classifiers are effective methods for sharing role and word information through the neural network. Regarding meaning composition, we apply a deep learning method for processing sequential data and sentence meaning. We further explore usage of attention mechanism to obtain a weighted meaning representation based on the Recurrent Neural Network(RNN) or word embeddings. We explore their performance at the phrase level and sentence level. Experimental results of perplexity indicate that the attention mechanism performs better at the phrase level than at the sentence level. Meanwhile, at the sentence level, the RNN is an effective method for sentence meaning representation. Furthermore, based on the evaluation of thematic fit difference, the attention mechanism applied in sentence level are also important for contextual meaning extraction.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

AN	Adjective-Noun
LSTM	Long-Short Term Memory Network
NLP	Natural Language Processing
BP	Back-propagation
BPTT	Back-propagation Through Time
CVG	Compositional Vector Grammar
DM	Distributional Memory model
DSN	Distributional Semantic Models
GRU	Gated Recurrent Unit
LDA	Latent Dirichlet Allocation
LM	Language Model
MT	Machine Translation
NN	Neural Network
NP	Noun Phrase
PCFG	probabilistic context free grammar
POS	part-of- speech tagger
PP	Perplexity
RNN	Recurrent Neural Network

SGD Stochastic Gradient Descent

SRL Semantic Role Labeling

VP Verb Phrase

Chapter 1

Introduction

1.1 Thematic role and role-fillers prediction

Thematic roles refer to a role that an entity expressed in the sentence. Usually, the task of thematic role labeling in natural language processing attempts to answer questions such as "who did what to whom?" [56]. To a large extent, the role that a noun or noun phrase plays in a sentence is closely related to the predicate verb. Usually, thematic roles can be defined differently. They can either be coarse-grained or fine grained (Geranmayeh [25]). Another concept related to thematic or semantic roles (Ferretti et al. [24]) is verb-specific thematic role concepts, where other thematic roles incorporate the verb-specific information. They proposed the concept that verb meaning and situation structure is related among the entities that commonly participate. Thematic roles concerned in this project include the role of *agent, patient, location, time* and *instrument*. When unspecified, we will also include the *verb* as one type of thematic role that the project will try to predict. All the other roles will then be marked as *other roles*. Here the concept of *agent*, refers to the participant which the meaning of the verb specifies as doing or causing something while *patient*, which the meaning of the verb specifies as doing or causing something[22] .

Thematic fit is the extent to which an entity fits a thematic role in the semantic frame of the event. The task of thematic role-filler prediction aims to find the most appropriate word for a target thematic role, given related context information or other thematic role information. In that case, the task of thematic role-filler prediction is also closely related with selectional preference. Whereas the latter refers to the word tendency of co-occurrence with some certain words in certain semantic classes. Either in the task of thematic role filler prediction or words' preference selection, the word's thematic role or semantic classes is not the only situation that must be taken into consideration, due to the fact that the process is a logical consequence of word's meaning (McCawley [44]). For example, the direct subjects for the word *eat* should be animate while it's direct objects should be edible. However,

according to (Sayeed et al. [62]), the drawback of previous studies on roll-filler prediction is entirely due to information about the "frequency of the word collocation, the syntactic dependences collected through corpus data and handmade grammars". Meanwhile how to effectively integrate the context information into the the task is the main focus of this thesis.

1.2 Thesis motivations

Given different context information *The **hardworking boy** goes to ??* and *The **sick boy** goes to ??*. The possible role-filler for *location* may be different. For example, *school* for the former sentence and *hospital* for the latter. In this project, given the input sentences, the thematic roles for each word or noun phrase, as well as the target role, we are expected to predict the role-fillers. So the research objective of this project is to effectively learn the role and word annotation of given sentences and combine the given information together for target role-filler prediction.

As mentioned in the previous part, there are similar tasks for thematic role-filler prediction and selectional preference learning. However, most of the literature focuses only on the content words of thematic roles, such as only the noun for the role of *agent, patient* and the verb for the verb predicate [20, 62]. Meanwhile, there isn't much research so far looking into the context information, especially the modifier of head noun of each thematic role. Furthermore, in previous researches, researchers tend to rely heavily on the Distributional Semantic Models (DSM), in which they extract the distributional information of high dimension vectors and similar words are defined based on the vector similarity achieved by the model. While one drawback of DSM is that the information for similar words extraction is heavily relied on the training data and the relation links defined in the model. This restricts the task, due to it being only effective on word pairs or triples [6, 40], and being rarely implied into context information of learning of a whole noun phrase or sentence.

In the recent years, the use of neural networks has dominated the learning of words' distribution using optimization [71]. The implementation of the neural network in the work of (Van de Cruys [73]) also shows its advantages in overcoming data sparsity problems. Thus, this project tends to explore the use of neural approaches for context information combination as well as for related thematic role-filler prediction.

1.3 Thesis structure

The thesis is organized as follows. Chapter two introduces the reader to the field of thematic role-filler prediction by introducing firstly the concept of thematic roles and their functions in constructing natural languages. Then it explores the various ways in which previous research has experimented with thematic role filler prediction, thematic fit acquisition and

selectional preference acquisition. After that, it also explores how the methods improved from role based methods to unsupervised machine learning mechanisms. Furthermore, it also summarizes the previous methods for learning context information of a given sentences/ noun phrases, and finally the necessities of adapting neural approaches for thematic role prediction or selectional preference acquisition, as well as the possible methods for neural approaches in thematic role-filler prediction.

Chapter three introduces detailed information about the proposed methods, of which there are two. First of all, we proposed the application of Recurrent Neural Network (RNN) on sequence of sentence processing. Later, we explain the attention mechanisms which have been proved effective in some other Natural Language Processing (NLP) tasks by assigning weighted interpretation for certain hidden states. Moreover, we also discuss how to solve the problem of wordrole information sharing within the model. When given different contexts, the thematic roles for certain word will change.The proper use of word-role embedding in this project can reduce the learning parameters as well as improve training efficiency of target role-filler prediction. Building a word-role parameter sharing matrix seems to be efficient way to solve this problem.

Chapter four describes how the experiments are carried out with a neural approach. The first part describes the training data and its preprocessing. The second part explores the word embedding used for experiment and the training model of word2vec. In this part, we also introduce the evaluation metrics. As there are limited resources for evaluation data for our project, we describe briefly how we construct our own data for thematic fit difference evaluation. The final part of this chapter introduces the training details and methods we used to prevent over fitting.

Chapter five shows the performance results of the different models we experimented with for context information learning and thematic-role filler prediction. It introduces the perplexity results and the models' performance on thematic fit based on human rating.

Chapter six consists of conclusions to be drawn from both of the approaches, and offers insight towards potential research directions for the future.

Chapter 2

Literature Review

This section contains the overview of topics and previous studies that have been conducted relevant to thematic role-filler prediction and context information learning. The aim of this chapter lies in twofold: first of all, we intend to provide basic information of the thematic role and thematic role-filler prediction, so to understand the methodology and discussion of the current work. Second, we will discuss the current research done in the related area and possible techniques for solving the problem, which may offer us the inspiration of the proposed method for our project.

2.1 Thematic roles and thematic role-filler prediction

In this part, we focus on the concept of thematic roles and their role-filler prediction. We will also provide information on the current research that has been done for target prediction.

2.1.1 Thematic roles and thematic role-filler prediction

Thematic roles usually refer to a set of semantic roles that a noun phrase has in relation to the verb (predicate) in the sentence. Given a sentence, usually all the noun phrases will fill in a semantic role for the given predicate. According to the relation between the predicate and the noun phrases, there are usually several different types of semantic roles such as *Agent, Patient, Location* and so on. Typical semantic roles will also include the adjuncts such as *Locative, Temporal, Manner, Cause* et al [79]. One of the basic Natural Language Processing task involved with thematic role is the task of semantic role labeling (SRL), which aims to discover the predicate-argument structure of each predicate in a given input sentence [79]. Usually, the realization of SRL, which is a supervised learning process, is closely related to the syntactic information contained in the sentences. Meanwhile, the information of SRL can be applied into practical tasks such as information extraction, question answering and so

on [8, 64]. Apart from that, the projects for thematic role-filler prediction, which are usually supervised machine learning tasks, are learned mainly based on the annotated semantic roles provided in the training data. In our project, as mentioned again, there are five types of thematic roles included, which refer to the role of *Agent, Patient, Location, Time, Instrument*. All the other semantic roles such as *Result, Goal* are all marked as *other roles*. Also, the predicate among each sentences are also regarded as the role of *verb*. The example illustrated below shows the information of the semantic role of each noun or noun phrase respectively with the predicate of the sentence.

- (The hungry boy)$_{[\text{Agent}]}$ eats$_{[\text{Verb}]}$ (cake)$_{[\text{Patient}]}$ (with a knife)$_{[\text{Instrument}]}$ (in the kitchen)$_{[\text{location}]}$
- (The sick boy)$_{[\text{Agent}]}$ took $_{[\text{Verb}]}$ (medicine)$_{[\text{Patient}]}$ (in the hospital)$_{[\text{location}]}$

Given the context information, such as the nouns and their thematic roles, the task is to prediction the noun head for target roles. As mentioned in 1.1, the task of thematic role-filler prediction is closely related to the role of the thematic fit evaluation or the word's selectional preference. Here the thematic fit refers to the extent to which the nouns or noun phrases fit a thematic role given the predicate of the sentence according to (Sayeed et al. [62]). The task of selectional preference concerns about the semantic restrictions that a word imposes on the environment in which it occurs[11]. Similar to thematic role-filler prediction, the exploration of a word's selectional preference is based on the observed frequencies of co-occurring words or phrase pairs together with their semantic role classes. This is why exploring the previous researches related to thematic fit evaluation or words' selection preference is quite necessary to help us understand the task of thematic role-filler prediction.

2.1.2 Distributional Model for thematic role-filler prediction

In the early stage, the calculation of thematic fit or selectional preference relies mainly on hand-crafted resources, such as Wordnet. In these types of methods, the class information is essential for preference selection. Usually the researchers focus on the the verb to class relation extraction and learning or class to class learning. In a study presented by [2], they propose a class to class relation learning integrated with the information extracted from Wordnet. Meanwhile,(Clark and Weir [17]) found a generalized level of preferential class for word selection according to the hierarchy structure of Wordnet. However, as (Ritter et al. [59]) commented, even though these methods produce a human-interpretable output, the classification models often suffer in quality due to an incoherent taxonomy, an inability to map arguments to a class (poor lexical coverage), and word sense ambiguity.

Recent methods for thematic fit evaluation or selectional preference modification focuses on the data driven methods. For example, to learn the selectional preference of given words, (Rooth et al. [60]) used an Expectation-Maximization (EM) clustering algorithm which is

based on the probability of co-occurring data in the corpus. Also, in the work presented by (Ritter et al. [59]), they propose an Latent Dirichlet Allocation (LDA) method. They focus on the latent topics and their topic distribution according to relations. They also combined the advantages of a class based method to produce human interpretable classes to describe the classes and their relations. The combination of the distributed method with the previous class based method increased their model's performance for selectional preference induction. Another topic on selectional preference learning model is proposed by (Séaghdha [63]), also with the adaptation of the LDA method. They assume the predicate in the sentence is associated with a distribution over semantic classes. This method enables them to draw multinomial distribution of predicates in sentences over all the semantic role classes as well as enables them to have multinomial distribution of semantic classes over the type. The usage of Bayesian techniques is also beneficial for learning probabilistic models of selectional preference and enables the effective learning of unknown words.

Apart from exploring the relation between thematic topics and their semantic classes, more popular methods recently conducted in this field are based on the distribution space of the words in corpus. To induce selectional preference, (Erk et al. [23]) proposes a method that uses corpus-driven distributional similarity metrics to learn the possible word co-occurrences. The use of distributional model is based on the assumption that words will share semantic similarities if they occur in the similar context[40, 53]. Basically, most of this research is based on Distributional Semantic Models (DSM) which extract the distributional information in high-dimensional vectors in order to define distributional/semantic similarity in terms of vector similarity [58, 59, 63].

The state-of-art DSM is proposed by Baroni and Lenci[6], named the Distributional Memory model(DM). The DM model is a multi-functional model, where the distributional information can be shared for different semantic tasks such as preference selection,word similarity judgments, discovering synonyms, or concept categorization. In their paper, they introduce the way of processing word pairs as triples. In these word-word pair triples, two of the sets are the two sets of objects while the third one denotes the relation between two objects. The concept of the triple information is also applied over text level, where the relation triple is between two sets of strings of content words in the text while the relation indicate the link between these words. This concept is also further developed into corpus level to present the words pairs and their relations among the whole corpus. A weighted matrix is also applied to learn the weighted distribution of tuples among the whole corpus. To train the weights for tuples, the DM model takes the form of an order-3 tensor, where two of the tensor axes represent words and the third axis represents the syntactic link between them. The application of this three-way tensor, especially the use of the third axis, allows the model to fully exploit the potential of corpus-derived tuples. Another feature for this three-order tensor is that the data stored in the tensor is called local mutual information (LMI), which is the basis for preference selection. By implementing this DM model, (Lenci

[40]) presents a computational model of the dynamic composition and updates the verb argument expectations using DM model. The work proves that DM can successfully predict the thematic fit between an agent-verb pair and a patient-noun argument of the same verb.

The description of selectional preference or thematic fit evaluation in this part shows that the research on these tasks mainly focuses on the frequencies of word pair co-occurrences and the target word's semantic roles or classes. Amongst this research, the DM model has a much better performance due to the fact that it's a corpus driven approach for analyzing the distributional similarities among words in similar or different classes. However, despite the good performance of the DM model in different NLP tasks, the results from these models depend heavily on how exactly the distribution space is defined, while having no principled way of optimizing the space [71]. Meanwhile the advantage of a neural network approach is that the model can be trained to optimize the distributional representation for the task [71] as well as to overcome the data sparseness problem[73] . Thus, a few researchers have explored the neural approach for thematic fit or selectional preference.

2.1.3 Neural Approaches for thematic role-filler prediction

Recently, neural networks have becoming very popular in Natural Language Processing tasks. Furthermore, (Bengio et al. [9]) also demonstrate the effectiveness of neural language models in the application of language modeling. However, only little research has explored the application of neural networks in selectional preference or thematic fit in literature.

One of the early applications of neural approaches in similar tasks of selectional preference is done by (Tsubaki et al. [72]). They propose a neural network model for co-composition of arguments/predicates. This model shows the effectiveness of adapting arguments/predicate meaning representations with the overall semantic information. (Van de Cruys [73]) apply two neural network architectures to multi-way selectional preference, where the neural models learn to discriminate between felicitous and infelicitous arguments for a particular predicate. To solve the problem of event detection, Dasigi and Hovy [21] introduce a novel technique using neural networks to model the representation of events as the composition of their predicate with the semantic information. His works proves the effectiveness of Recursive Neural Network (RNN) in semantic composition based on semantic tree structured data.

An inspiring research that relates closely to our project is presented by [71]. Their work focuses on learning the representation of events and related thematic roles in order to calculate the probability distribution over the possible role filler of specific missing roles. As the predictability of these words depends heavily on the relationship of these words to other nouns and verbs in a large quantity of the corpus, they present a neural network model to learn the words and their role embedding. Besides, the model is compositional with that fact that it can handle several role-fillers at the same time and thus predicts the role filler

from a combined representation of the co-concurrence of participants in the same event or the predicate itself.

Based on the same data from (Sayeed et al. [62]), (Tilk et al. [71]) present a novel Neural Network (NN) model for the probability distribution generation. The structure of their NN model includes a non-linear hidden layer and a *Softmax* output layer. In their task, there are many problems they need to consider. First, in the embedding layer, the word embedding should be different depending on the context, as the word in *agent* role might be different from that in a *patient* role. Second, the classifier layer should be different for each target role, as the target role filler for various roles will be different depending on the context. In that the case, the two sets of parameters W and W' for the two layers should be different. Keeping a separate model for the input and target role pair raises the problem of a lack of parameter sharing matrices. Their solution is to share the role-specific embedding and classifier weights together in the NN in order to to enable the combination of all input-target role pair models into a single one. Inspired by work from Memisevic and Hinton [47], Sutskever et al. [69], they proposed a method of forming a 3-way tensor for the embedding and classifier layer respectively. Each tensor consists of three factor matrices which will alleviate the efficiency and solve parameter sharing problems. To train the model's parameters with the factored tensors, they also report the effectiveness of using AdaGrad as the gradient algorithm.The drawback of this approach is that the number of parameters grows rapidly as the vocabulary size increases. Thus, tensor factorization is used to reduce the number of parameters. Generally, they have implemented two type of NN: the incremental mode,l which is a Recurrent NN, can include the information of word order and the content words and their roles. The other one is the non-incremental model, which adds all of the input of the context together into the word embedding layer.

In this section, we introduce the distributional approach and neural approach for a related task. As mentioned above, both the DSM and NN approaches show promising results for the NLP tasks of selectional preference and thematic fit. However, research in both the traditional DSM model and the NN approach [40, 62, 71, 73] build their semantic spaces by splitting the verb arguments or the noun head of arguments into separate vector dimension. Thus most of these frameworks are based on the interaction between topics or the head noun of each semantic class with its predicate. Even though some of the researches such as framework presented by Lenci [40] can handle multiple word pairs or triples preference selection, the context information referring to the participants is usually ignored. Although, examples illustrated in 2.1.1 show that the context information or the modifier for each content phrase such as *sick boy* or *hungry boy* may also influence occurrence of nouns or verb in other semantic classes under the condition of a similar predicate. Thus, it's also necessary for us to explore the influence of context information on thematic fit or selectional preference.

2.2 Context information Composition

2.2.1 Context information used in selectional preference or thematic fit

In the previous section 2.1, the thematic fit models for a distributional approach or a neural approach tend to span a sequence of words or a word window into a set of words in order to separate each of the words with their predicate. Though all these models show good performance in word pair or triples selection, they will also lose the valuable context information for word inter-correlation. In that case, another trend for the NLP task on selectional preference is to explore the contextual information in the DSM model.

The first attempt to do so comes from (Ruiz-Casado et al. [61]), who propose the concept that the semantic similarity between two word-forms is based on their similarity between context. This concept basis enables them to use context information for hyponymy or synonymy detection. For example, in the sentences below, the *woman* is defined as the hyponym of *queen*.

- a. The Prime Minister honored the queen with his presence.
- b. The Prime Minister honored the women with his presence.

Under the condition of sharing same context information, if the word *queen* in sentence a can be substituted by the word *woman*, they will define the latter one as the hyponymy of the former. So, in order to get the context information among word pairs, they introduce this unsupervised learning method. And they present a context-window overlapping algorithm to collect a list of contexts where *word1* appears. It counts in how much of this context shows in the cooccurence to substitute *word1* with *word2*.This methods shows good performance in the synonym detection test which outperform the approach based on history information without context.

Followed by (Ruiz-Casado et al. [61]), (Agirre et al. [1]) introduce a context-based model to study word similarity and relatedness. Similar to the previous research, they determine the word similarity based on the context similarity between two word sets. This model adopts a richer context representation by considering entire word window contexts as features, while keeping the same computational vector-based model. Though its performance for the similarity evaluation task is much more compatible than the existing methods, this approach suffers from a very high-dimensional feature space resulting in data sparseness problems [46].

Later, (Melamud et al. [46]) introduces a generic distributional similarity scheme to effectively learn the joint contexts. Also based on the concept that similar words can be substituted in the same context,they calculate the probability of words given the similar

context information. To overcome the problem of data sparseness, they use the Kneser-Ney n-gram model [36] and their model outperforms the existing DSM in their tests. Though the implementation of an n-gram language model is an effective method to define word similarity or selectional preferences, it is challenging to apply it into our project based on the fact that each word is annotated with its semantic role information and the context will change according to not only words information but also their semantic classes.

Another model for incorporating context information in a DSM is proposed by (Chersoni et al. [13]). In their paper, they introduce a model of extracting verb context information by syntactic dependencies (*object,subject, complement*). Like previous research which mainly focuses on the linear window size information, the method of considering syntactic dependency is more effective , especially for verbs and their context information extraction. The models are later tested on the verb similarity tasks and shows that the joint contexts method is comparable to or even better than single dependency methods. However,taking the whole sentences' information into consideration, the syntactic structure based method for getting context information between other semantic role classes, such as the relation information between location and agent in the sentence, is much more complex.

On the one hand, based on the above research, we may assume that the context information compositing model may also influence the predictions of thematic role filers tasks. [71] only take the single noun as the head of input for the semantic roles, it's necessary for us to combine the context information of noun phrase in the role-filler predication task.

However, on the other hand, most of the DSM methods used in this previous research use abstract information from a linear word window or based on the syntactic structure. Though these machine learning techniques have shown promising performance for context information composition, their limited window size is not able to handle long-dependency information. Thus, most of this previous work only focuses on the selectional preference or semantic fit of one or two words pairs related to the predicate. Meanwhile, integrating the long dependency information for context information between semantic roles such as *agent* and *location* in the case of *the sick boy is taken by his parents to the hospital* is still an open issue. Since neural approaches show advantages in handling large amount of data as well as information composition and feature extraction for long-dependency sentence, we will try to explore a neural approach for our task.

2.2.2 Possible methods for context information composition

As the goal for our project is to investigate the influence of context information in role-filler predication, the input will be a noun phrase instead of a single noun from the arguments of the verb. The method used in the previous work, which takes the word's position in vocabulary as one-hot vector input, does not satisfy our requirement for embedding. For this reason, word2vec is a more appropriate way for us to handle words embedding in the phrase. The

major problem for our project is to achieve effective embedding of the context information. With this goal in mind, there are some specifications that are necessary when developing the system.

First of all, in noun phrases, not all the words share the same importance as role filler predication, due to its word type or semantic/syntactic relation with the head. For example, in the sentence ** *will affect the boy's system*, in the NP of *patient*role, words such as *system* and *the* won't be as important as *body* for the prediction of *virus*. Thus, we need to put more weights on the words which may have a decisive effect on the predication.

Secondly, besides the modifier or compound word which may affect the meaning of head, there are long dependency issues we need to take into consideration. As there are many long sentences in any corpus, it is also important for the model to capture the necessary information from the word whose position is far away from the head noun. In the example "the boy who is sitting on the bus, and playing with classmates will go to **". Here, the word *classmate* in the noun phrase of *agent* may be a good hint for the prediction of *location* as *school* instead of *hospital*.

Bearing this in mind, the priority issues during the method exploration for our project is to satisfy both finding weighted embedding of words among sentences and being able to handle information with long-term dependency. In the following part, I will discuss the possible methods we may use for noun phrase meaning representation and sentence meaning composition. We will also discuss the possible reasons for why they are preferred or not.

Straightforward methods for sentence meaning embedding

As suggested from the previous research into DSM, the multidimensional word vectors is quite effective in NLP tasks as they represent the words' meaning by distributional information and reflect the semantic similarities. Though the word vectors are effective for thematic fit or selectional preference based on single nouns or verb in the sentence, their application on sentence information composition should be investigated with effective methods.

The most common method for vector combination is averaging all the vectors in the sentences. However, the way of simply averaging word meaning is not sensitive to word orders according to (Landauer and Dumais [38]).

- a. It was not the sales manager who hit the bottle that day, but the office worker with the serious drinking problem.
- b. That day the office manager, who was drinking, hit the problem sales worker with a bottle, but it was not serious.

As examples shown above from the research of (Landauer and Dumais [38]), both sentences include the same sets of words but present different meanings. Thus averaging the vectors in the sentence may not be an effective way for information learning. Apart from that, these examples also indicate the importance of syntactic structure for semantic information learning, because the difference between two example sentences is realized by different syntactic order of the same sets of words.

(Mitchell and Lapata [54]) presented some other straightforward but potentially effective methods for sentence or phrase meaning composition. One of the most frequently used methods for sequences of words meaning representation is addition. The advantage of addition for sequence is that it will keep the vector in the same dimension. However, the drawback of addition is its independence from sequence orders and will not capture the meaning difference with words arranged with different syntactic structure. Instead of simply adding all word vectors in the sequence for meaning representation, (Mitchell and Lapata [54]) propose an weighted score for words with different syntactic structure. Another method they proposed for sequence meaning composition is multiplication among word vectors. They apply the proposed methods into sentence similarity detection tasks and the results show that the way of multiplication function is superior to the additive one.

Compositional Phrase Embedding

Compositional phrase embedding refers to the computing of phrase embeddings from words embedding by using various kinds of compositional functions. Most of the common approaches for it is to use pre-defined composition operators similar to the method mentioned above. In the research presented by (Erk et al. [23]), they presented a phrase meaning composition methodology using vector computation. Whereas the contribution from their method compared with other research is that, the vector they used combines both the word's information and the expectations of the word. For example the vector representation of word *catch* is combined with its own lexical meaning together with the *object* information of words such as *he, fielder,dog* and with the *subject* information with *ball, cold, drift*.

Another related work is presented by (Milajevs et al. [52]), where they introduce a tensor based approach for sequence meaning composition which can be applied into any of the compositional tasks. Their proposed method is based on the generalization of the notion of vectors, named *tensors*. They introduced a vector v as an element of an atomic vector space V, while a tensor z is the element of tensor space of the addition or multiplication of several atomic vector spaces such as *V,W or Z*. Then, during the experiment, the meaning of nouns are vectors and meanings of the predicates are tensors. The meaning of the string of words is obtained by applying the compositions of multi-linear map of the tensors to the vectors.

Under the topic of phrase meaning composition, some researches focus on the adjective-noun(AN) meaning composition[54, 7]. (Mitchell and Lapata [54]) introduce two ways

of AN composition: addition and multiplication. In both methods, adjective and noun are regarded as word vectors. For addition, the meaning composition is realized by adding the linear projection of two vectors (with weighted matrices for each vector). At the mean while, multiplication refers to the adaptation of weight tensor for projecting the two vectors into their meaning composition. Similar to this method, (Baroni and Zamparelli [7]) regard adjectives in the attributive position as linear function of noun tensors in AN meaning composition. Therefore, they derive the AN meaning by multiply the noun vector with the weight matrix of adjectives. Though these researches focus on AN instead of noun phrase (NP) meaning composition, their effectiveness further prove the necessity of using word vectors in meaning composition.

As most of the compositional modes can not make good enough use of the weighted information of each word to fulfill our task requirement, we won't talk about the details of those methods. However, one of the models proposed by (Yu and Dredze [78]) might be relevant to our task. Their model learns composition functions that rely on phrase structure and context and can produce a weighted summation of embeddings of component words. For example in an NP which has a flat structure, all the words would modify the head nouns, therefore it enables their model to favor the head in the compositional embedding. Apart from that, their model can be trained with both unlabeled and labeled data, and the linear conformation function makes it fast to train.

Even though the model they proposed has lots of advantages, which may be helpful for our task, there is one drawback: they treat every sentence or phrase as compositional. While it may not be suitable for non-compositional phrase such as idioms. The ideal method might be building a model for a general phrase , which has been tried by ([77, 29]). However,as both of these two researchers are trained on only transitive verb pairs in a bigram model, they won't be able to handle multi-gram phrases in our case.

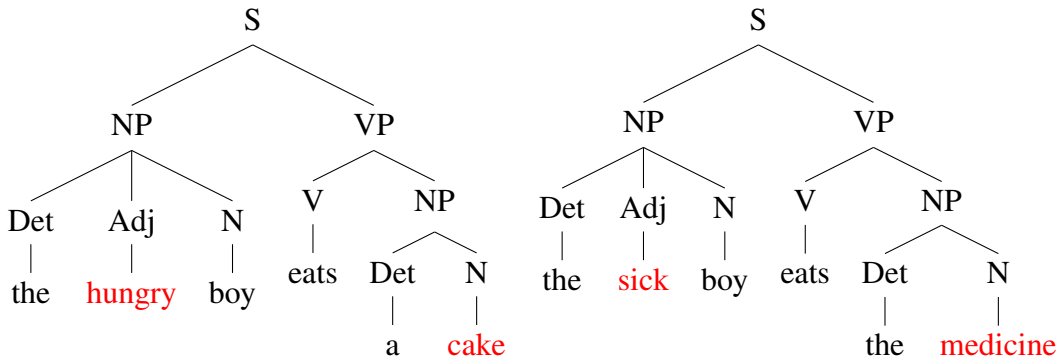
Apart from phrase meaning composition, (Coecke et al. [18]) proposed a mathematical framework for sentence meaning composition. To capture the meaning of each word in the sentence and combine them informatively, they involve two categories of natural language information: vector space for words' meaning representation, and Pregroups for grammar structure of each sentence. For each sentence, each word is assigned with a syntactic type and vector space based on the syntactic information. Later the sentence meaning is computed by syntactic reduction based on these tensor space. There are two ways of syntactic reduction for positive or negative sentences and their proposed methods cover various types of sentences meaning composition. However, as the proposed meaning composition depends on the syntactic structure of sentences, taking both syntactic and semantic information for meaning composition will make our project much more complex.

Recursive Neural Network

In natural languages, the syntactic roles of language tend to be recursive, and sometimes a noun phrase in sentence may also contain some other noun phrase such as in the example *the hungry boy who is super hungry right now .....* In the field of natural language processing, the recursive neural network refers to the NN that usually applies the same set of weights recursively over a syntactic or semantic structure. This enables it to successfully learn sequence and tree structures in natural language processing.

As mentioned in the previous section, the syntactic structure plays an import role in semantic information integration. Thus, it is necessary for us to explore the possible approach involving integrating syntactic structure of sequence of words into a composition meaning. The syntactic tree illustrated below shows its nature of recursion. furthermore, with the help of recursive NN, the context information of *sick boy, hungry boy* and the corresponding *patient* role-filler *cake, medicine* in the examples shown below can be easily focused on the *NP* of the sentence and the *NP* under the tree node of *VP* (verb phrase).

The Recursive NN has proved its efficiency in semantic tasks such as semantic composition [67, 21]. Two types of tree structures have been applied in the previous study for semantic meaning composition. (Dasigi and Hovy [21]) introduce the semantic relation based tree structure for anomalous event detection while (Socher [66] Socher et al. [67]) choose syntactic tree structure and sentiment tree structure respectively for semantic meaning composition.



In the thesis presented by (Socher [66]), he proposes an Recursive NN based model to learn features and phrase representations which can also handle long-term dependency and un-seen n-grams. This multi-functional model is called a Compositional Vector Grammar Parser (CVG), which is able to both parse the syntactic structure of a sequence and present the phrase compositional meaning. The multi-functional model makes use of both probabilistic context free grammars (PCFG) and Recursive NN structure, which means it can capture the discrete categorization of phrases into NP or VP as well as capture fine-grained syntactic and compositional semantic information among phrases and words [66]. Also, the combination of PCFG and Recursive NN enables the model to deal with ambiguity between sentences

with the same syntactic structure. Examples of ambiguity despite identical syntactic structure sentence is listed below.

(1) They ate cake with forks.

(2) They ate cake with dogs.

The realization of phrase meaning composition in CVG is also based on word vectors. According to [66], the weighted score for each node in the tree structure that is combined into the sequence meaning is determined by the categories of the child constituents. This child constituent based weighted score composition allow different composition functions when combining different types of phrases.The detailed explanation of Recursive NN sequence meaning composition is shown in the figure 2.1 below.

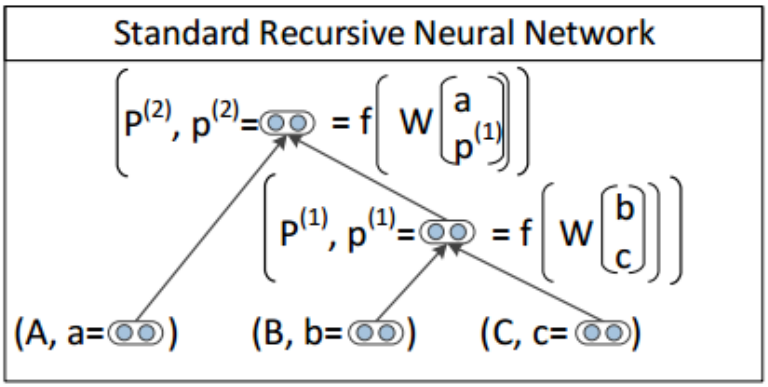


Fig. 2.1 Syntactically Untied Recursive Neural Network
Socher [66]

As demonstrated in Figure 2.1, the first parent node is calculated as :

$$p^{(1)} = f(W^{(B,C)}[{}^{c}_{b}]) \tag{2.1}$$

where $W^{(B,C)}$ is a weight matrix and its value is determined by the child constituent b, c. Apart from calculating the parent node's vector, a composition score is also calculated based on the Recursive NN and the PCFG. Here, the PCFG is used to define the log probability of combining the two child constituents in syntactic level. The score for $p^(1)$ is listed as:

$$s(p^{(1)}) = (v^{(B,C)})^T p^{(1)} + logP(P_1 \to B,C) \tag{2.2}$$

where the probability of $P_1 \Rightarrow B,C$ is referred from the PCFG. Then, the calculation of the next parent vector is based on its child constituent as well as the phrase vector $p^{(1)}$:

$$p^{(2)} = f(W^{(a,p_1)}[{}^{a}_{p^{(1)}}]) \tag{2.3}$$

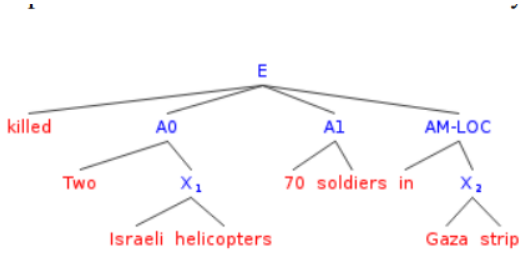
The score for this parent node composition is calculated as :

$$s(p^{(2)}) = (v^{(A,P_1)})^T p^{(2)} + logP(P_2 \rightarrow A, P_1) \tag{2.4}$$

Later, the model is trained based on the combination scores of each parent nodes in the sentence and is assigned with a composition matrix, where it is able to assign more weight scores to the head nouns rather than the adjective or determiners in the sentence.

Although Recursive NNs shows potential in combining sequence information with weighted scores among different categories of words in one sentence [66], we can not directly adopt it into our model as there is thematic role information missing from the information composition. Therefore we need to further explore the application of Recursive NNs in the field of anomalous event detection, which interprets the words in a sentence based on their thematic roles with a Recursive NN architecture.

The application of Recursive NNs in the field of semantic information detection is presented by (Dasigi and Hovy [21]), where they adopt an semantic role structured Recursive NN architecture. To learn whether an event used in newspaper titles is anomalous or not, they use unsupervised learning methods to integrate the semantic role-fillers' information in the sequence and then learn the event-semantic role-fillers composition with a supervised method. To learn the thematic role-fillers information in the sentence, they introduce an event tree, where the structure is formed based on the noun phrases' semantic role with the related event in the sentence. Below, the figure is an example of event tree for the sentence *Two Israeli helicopters kill 70 soldiers in Gaza trip* [21].



The method for semantic compositional model consist of two parts: the first part denotes to the argument composition in the sentence referring to the related event while the second part includes the event composition with the related arguments. To realize the first task, all the parameters for node composition among the sentence are the same and then contrastive estimation methods are adopted in order to learn the compositional scores in this way:

$$argminJ_{arg} = argminmax(0, 1 - s + s_c) \tag{2.5}$$

where s refers to the score of the composition of the entire argument while s_c refers to a randomly replaced word in the argument at each time. Later, the event composition is

calculated by the combining the argument score and the event representation together with the label of the event (being normal or not with the given context of argument). The evaluation result shows that their proposed method can be applied successfully into anomalous event detection.

The use of event tree based Recursive NNs for event detection may offer some inspiration for our task. However, during their training, the detection of being normal or anomalous is only a binary classification, while the classes for role-filler prediction in our task range into the whole vocabulary of the training corpus, as each word appearing in training corpus may be a possible role-filler for a given context. Besides, during their second part of composition, namely the "event composition", the representation of predicate information and its label plays a dominate role in meaning composition. However, in our project, given different context, the role-filler of *verb* might also be expected. Thus, the proposed method by (Dasigi and Hovy [21]) can not effective get all sentence meaning composition when the verb information is missing.

From the previous application of Recursive NNs in semantic meaning composition and event type detection, the Recursive NN shows advantages in assigning a higher score for relation weights such as attributes and lower score for the weights of determiners. The nature of Recursive NNs in this field may enable us to get the determinant information for target-role filler prediction. However, there are several problems we need to take into consideration with this method. First of all, we need to extract all the words included in noun phrase, parse them with the dependency parser and then update the the syntactic relation information in our training corpus. Considering the comparatively large size of the training corpus, this type of data annotation will be very time-consuming. What's worse, the input of our model will require not only the word2vec and the word's semantic role but also the relevant syntactic relation label among each word. The several different types of input will make the architecture of model more complex and may easily lead to over-fitting. Apart from that, determining relation weight matrices for word combinations will require manual work for rule construction, which is not scientific at all, as one certain type of relation words may require different weights under various contexts.

2.3 Summary

In this part, we introduced the concepts for thematic role-filler prediction and the previous work conducted for it or related tasks such as selectional preference detection or thematic fit prediction. These works indicate the efficiency of using multidimensional distributional word vectors in finding a similar or related word. Following on from their work,we will also use word vectors for model learning and target word prediction.

Apart from that, we have also discussed several possible methods that have been explored for context information integration or sequence meaning learning. Among all these methods, the most straightforward way is word (or word vector) addition or multiplication. However, as (Landauer and Dumais [38]) mentioned, these methods are not sensitive to word orders and may have difficulty in distinguishing sentence ambiguity. Thus we may only use word addition as a baseline for our experiment. Even though there are compositional phrase embedding model presented from the previous work by (Milajevs et al. [52], Yu and Dredze [78]) and so on, most of the models will focus still on the verb predicate and the head nouns of its semantic argument, which are usually bigram or trigram models. Thus, these methods can not be applied into our task for sentence meaning composition. Among the neural network approaches, the model architecture proposed by [71] offers us an inspiring idea for effectively sharing words and their thematic role information during model training. Followed by their description, we will also use an factored word-role embedding for individual word information learning. Research presented by [66, 21] shows the potential of Recursive NN phrase meaning composition for anomalous event detection or meaning composition. Meanwhile, in (Dasigi and Hovy [21])'s work, the meaning composition of phrases is dominated by the event meaning. In our project, we should assign equal importance to the thematic role fillers in the given sentence and will also use the verb as the target word for training. In this case, the event or verb dominated composition method is not suitable for us. Moreover, [66]'s method requires both syntactic and semantic information for learning, which will increase the number of parameters for the model to learn.

With this in mind, in the next chapter, we will introduce the proposed method for learning thematic role and role-filler embedding together with context information in a sequence.

Chapter 3

Proposed Approaches

In this chapter, we will give a detailed description about the approaches to be adapted in our project. As discussed in the previous chapter, there are two important issues we need to figure out. First, we need to explore how to combine sentence/phrase information properly with a given sentence. Second, the model should effectively combine information about the word and it's corresponding thematic roles. The second concern is based on the fact that given different context, *the **dog** barks* and *the boy likes the **dog***, the thematic role for *dog* changes. Thus, to solve the first problem, we propose a recurrent neural network architecture for weighted sentence/phrase information learning. While considering word-role information sharing, we adapt a factorized-tensor suggested by (Tilk et al. [71]).

3.1 Recurrent Neural Network for Contextual Information

3.1.1 Simple Recurrent Neural Network

A Recurrent Neural Network (RNN) is another type of artificial neural network which features a model consisting of a sequence of inputs with a dynamic memory structure. As it applies its function iteratively on a sequence of inputs, it takes the input as a series of time step $w_1, w_2, ...w_n$ and processes the sequence of inputs one by one. At the same time, it can also take information stored in the previous steps into account. Specifically, for each time step, the RNN will update the hidden state (h_i) of the current input together with the hidden state (h_{i-1}) from the previous time step. Here the hidden state refers to the transformed word vector (word embedding) and a weight matrix added with a bias that also takes the previous hidden state into account. The unrolled architecture of the RNN is shown in figure below[Colah].

At time step t, with the given input word vector w_t, the current hidden state is calculated with the following equations:

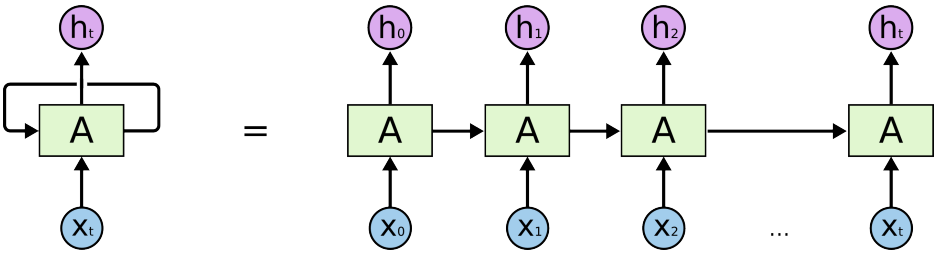


Fig. 3.1 An unrolled recurrent neural network[Colah]

$$\begin{aligned} s_t &= f(Ww_t + Us_{t-1} + b) \\ y_t &= Vs_t \\ P(w_t|w_1..w_{t-1},\Theta) &= softmax(y_t) \end{aligned} \tag{3.1}$$

where $\Theta = \{W,U,V,b\}$. W,U,V are matrix parameters, b is the bias, w_t is the input word vector and s_{t-1} is the hidden state at time t-1. During training process, the weight matrix W,U,V are shared across time steps. Here f refers to some non-linear activation function such as tanh or ReLU. The starting state s_0 is set to 0 to denote the initial state of the memory. If the target of this RNN is to predict the next word given the previous word history, then the probability of the next word is calculated with the $softmax$ activation function. The equation for $softmax$ is shown below:

$$p(x|h) = \frac{exp(y_i)}{\sum_j exp(y_j)} \tag{3.2}$$

So the probability of each word w_i given the context h is estimated by normalizing all values in y.

The reason for choosing an RNN for our sentence learning project is based on its ability to maintain a hidden layer of neurons with recurrent connections to their own previous values for an input sequence. So given sentence *the hungry boy is eating ??* , if we are trying to predict the role-filler of *patient*, the previous information provided by the RNN is sufficient to get the target word. The application of RNN's in language modeling and NLP tasks has already shown their good performance [49, 51]. Work presented by (Tilk et al. [71]) also proves the potential of using RNN's for sequenced information processing.

The training of a simple RNN is done by Back-propagation Through Time (BPTT). Here back-propagation (BP) is used to calculate the error contribution of each neuron after a given batch size of data. The target for neural network training is to minimize the error between predicted result and the real output, so the idea behind BP is to use an optimization algorithm for finding the weight matrix with stochastic gradient descent (SGD) of the loss function or error. As RNN processes a sequence of input data, the concern for BPTT is that the weight matrix for the NN at each time step is different. For a traditional Feed Forward NN, during

the training process the gradient is back-propagated to the previous layer with the recurrent connection along the NN. BPTT differs from BP in a feed-forward NN in that the gradient of BPTT can be propagated into two ways: it can be propagated through the hidden state to the input layer of data or it can go back to the previous step on the same layer. (Here feed-forward NN is a traditional NN architecture in which the information for each input moves forward and there is no interaction between inputs.)

Despite the advantage for learning time step information for sequenced inputs, RNN also has problems with training. Usually, there are two types of problems during RNN training, namely vanishing gradient and gradient explosion [32]. These drawbacks also preclude RNN from retaining memory of more distant, long-dependency steps. During BP through time, the gradient can grow exponentially large while sometimes it will decrease dramatically to zero. Both of these problems cause difficulties with training. The problem of gradient explosion can be solved by clipping the gradient during training, while to solve the second problem of vanishing gradient, there are several methods proposed in the previous research. In the work proposed by Mikolov et al., they extend the basic structure of an RNN with an additional *feature layer*. The *feature layer* is connected to both the hidden layer and the output layer of the RNN and contains complementation information for their specific task. The equation below explicitly explains how the structure of the model was modified.

$$\begin{aligned} s_t &= f(Ww_t + Us_{t-1} + Ff_t + b) \\ y_t &= Vs_t + Gf_t \\ P(w_t|w_1..w_{t-1},\Theta) &= softmax(y_t) \end{aligned} \tag{3.3}$$

Besides the vector and weight matrix mentioned in the previous equation for the simple RNN structure, here F, G refer to weight matrix for the additional feature layer. Apart from adding an additional feature layer, (Le et al. [39]) introduce a simple modification: to play a initialization trick by using Rectified Linear Units (ReLU) as the activation function. However, these extensions are all based on a simple RNN architecture. Even though these methods have been proven to make RNN's acquire additional information, they cannot be trained effectively with BP for complex RNN models. The most successful method applied to sequential modeling in general are Long-Short Term Memory (LSTM) networks which include gate cells to explicitly deal with the vanishing gradient problem. Thus in the following part, we will explore the mechanism of LSTM for acquiring long-dependency sentence processing.

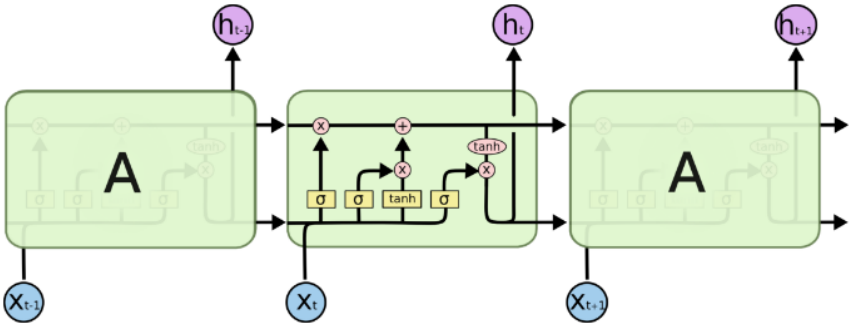


Fig. 3.2 The repeating module in an LSTM contains four interacting gates

3.1.2 LSTM

Long-Short Term Memory network (LSTM), proposed by (Hochreiter and Schmidhuber [34], is one specific type of RNN, which can overcome the problems that gradient descent has. Unlike a simple RNN, LSTM's integrate a linear memory unit into the hidden state so that the gradient can flow smoothly during the BPTT with the help of a memory cell. The application of these memory gates depends on the current input w_t and the previous hidden state h_{t-1}.

Normally, there are four gates in a single LSTM neuron. The forget gate f_t is used to directly control the memory flow c_t and sever the connection with the previous steps, the input gate i_t decides the amount of input to be incorporated, the output gate o_t controls the amount of memory flow to be produced for the task, and finally the candidate memory unit $\tilde{c}_t$ contributes to the current memory flow. After getting $\tilde{c}_t$ for the current candidate memory, we then decide the new information to be kept in the new memory cell. The new memory cell is updated by the combination of the candidate memory cell and the input gate. At the same time, the forget gate is also employed to determine the amount of information that can be updated from the previous memory cell. Finally, the new hidden state h_t is updated with the information stored in the current memory cell together with the output gate. The overall illustration of LSTM is shown in Figure 3.2 [Colah]. The computational details for the gates that make up the LSTM are shown in the equations in 3.4

$$
\begin{aligned}
f_t &= \sigma(W_f w_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma(W_i w_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma(W_o w_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= tanh(W_c w_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot \tilde{c}_{t-1} + i_t \odot \tilde{c} \\
h_t &= o_t \odot \sigma(c_t)
\end{aligned}
\tag{3.4}
$$

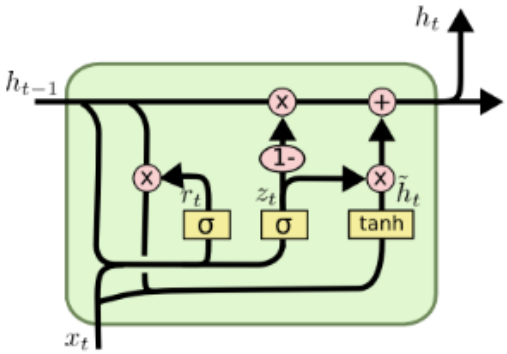


Fig. 3.3 The interaction among single GRU layer

where $\Theta = \{W_f, W_i, W_o, U_f, U_i, U_o, b_f, b_i, b_o, b_c, W_s, WE\}$. All W's and U's are matrix parameters, b's are bias parameters, WE is the word embedding matrix, w_t is the word embedding (from the WE matrix) at time t, and h_{t-1} and c_{t-1} are the hidden state vectors and context vectors at time t-1, respectively. The symbol $\odot$ represents element-wise multiplication. σ refers to the element-wise application of the sigmoid function on the input vector, which produces an output vector whose elements are in the range $(0,1]$.

This demonstration of an LSTM shows that, unlike a simple RNN, where the current hidden state is overwritten at each time step, the LSMT can capture the previous and current memory through the different functional gates within it [16]. With this functionality, if some important information is detected at the early stage of an input sequence, the LSMT can capture this information and retain it as a long distance dependency. Thus, it can be applied for context information learning over a long sentence in out project.

LSTM's can be efficiently implemented by computing all gates in one single matrix multiplication, then applying the activation functions on different parts of the output. In practice, there are several variations of LSTM implementation. One derivation of an LSTM is the Gated Recurrent Unit (GRU) [14], in which the gates are implemented differently than the original LSMT.

Unlike traditional LSTM's, a GRU combines the input gate and forget gate into a single update gate. Further, the cell memory state and hidden state are also merged together. There are two main states in GRU. One of these is the unit update state z_r, which determines how much information the unit used to update its information. The other, r_z is the reset state. If r_z is set close to zero, it will allow the current hidden $\tilde{h}$ state to forget all the previous information. An illustration of a GRU is shown in Figure 3.3 [Colah]. The computation details for the gates in a GRU are shown in Equation 3.5 below.

$$
\begin{aligned}
z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) \\
r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]) \\
\tilde{h}_t &= tanh(W x_t + U_z(h_{t_1} \bigodot r_t)) \\
h_t &= (1 - z_t) h_{t-1} + z_t \tilde{ht}
\end{aligned}
\tag{3.5}
$$

The terms representing the GRU in this equation are similar to those in Equation 3.4. However, there are apparent differences between them. Unlike the LSTM, which has four gates (forget gate, input gate, output gate and memory gate), there are only two gates in a GRU. Here o_t is the update gate and r_t is the reset gate at time step t. $\bigodot$ signifies element-wise multiplication.

From Figure 3.2 and Figure 3.3, it is easy to recognize the similarity between LSTM's and GRU's; both can retain previous content and add new content to the hidden state. In general, the four gates in an LSTM and the two gates in a GRU provide many advantages for NN training [16]. Firstly, due to the forget gate of the LSTM and the update gate of the GRU, important features from previous states are maintained and remembered in the current state. There features again allow them to deal with input sequences that have long dependencies. Moreover, the existing gates also provide "shortcut paths that bypass multiple temporal steps", allowing error to be back-propagated easily without vanishing too quickly [16].

Empirical research from (Bahdanau et al. [5]) shows that there is not much difference in performance between these two types of NN. However, GRU has a simpler structure and can be trained faster than an LSTM. We will try both approaches for this project.

3.1.3 Attention Mechanism

The attention mechanism is mainly inspired by the mechanism of human vision. Given a visual input of scenery, a human tend to focus his or her *attention* on specific parts of the scenery to analyze it, as opposed to analyzing the whole scene. The task of summarizing the content of an image is also known as image captioning. (Xu et al. [75]) introduce a neural approach with an attention mechanism for the task of image captioning. Recently, attention mechanisms have also been applied to NLP tasks. Among these applications, popular and successful ones are related to the task of Machine Translation (MT).

The standard neural approach usually adapted in MT is sequence-to-sequence learning, in which the model takes a sequence of words as input with an *encoder*, which then transforms them into an internal representation. Later, the new representation is transformed by a *decoder* into anther sequence of words in the target language. Both the *encoder* and *decoder* are RNN's. For a sequence of input $(x_1, x_2, ...x_n)$, the encoder will produce a sequence of hidden states $(h_1, h_2, ..., h_n)$. Usually, with an RNN such as an LSTM, the hidden state will contain information primarily from the current input as well as information for the previous state of the sequence. The attention layer added on top of the encoder is a feed-forward layer,

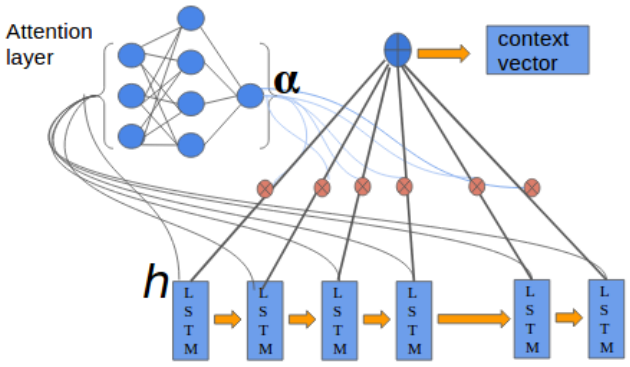


Fig. 3.4 The attention mechanism added on top of LSTM layer.

which enables the system to know how much information it should take from the input to get the output of the decoder. This is realized by assigning a weighted score α to the input of the decoder, which is the output of the encoder h_t at time step t. Thus, given contextual information from the encoder side c_t, and h_t, the attention score for the decoder is computed as:

$$\begin{aligned} e_t &= f(W_s[c_t, h_t]) \\ \alpha &= softmax(e_1, e_2, ...e_t) \end{aligned} \tag{3.6}$$

where $i = 1, 2, ...T$. Here f is a linear or non-linear function. The attention score α indicates how much the attention is focused on each of the input words for the decoder. Then the context vector c is calculated by using these weights and the hidden state of the encoder h:

$$c = \sum_{T}^{t} a_t h_t \tag{3.7}$$

Here the equations shown for e_t are generic and there is much variation for different NLP tasks [42, 57, 76, 80]. Unlike Neural MT, which takes sequences of words as input and produces another sequence of words as the output, our task takes a sequence of words as input but aims to predict a single word as the target. As with the context vector c from the encoder, the attention representation in our task is modified as:

$$e_t = tanh(Wh_t + b) \tag{3.8}$$

Then the attention score α for each word in the sentence and its context vector c is calculated as the previously shown in Equation 3.6. As mentioned before, the attention layer is usually added on top of a LSTM or RNN neural network layer. The architecture of a Attention mechanism added on top of LSTM, as used in our project, is shown in Figure 3.4.

Taking the hidden state h_t of LSTM output at time step t, the attention score α is calculated within a feed-forward layer. It is then multiplied with the corresponding hidden state h_t to get the weighted representation for each h_t and then the context vector for the input sequence is calculated by summing all of the weighted hidden states together.

3.1.4 Summary

In this section, we introduced the architecture of RNN, LSTM, and an attention mechanism for learning contextual information for our project. The main benefit of an RNN compared to a simple feed-forward neural network is that it can process a sequence of input sentences and capture contextual information. The recurrent connections in an RNN allow the model to 'memorize' the previous inputs and save them in the model's internal state. Having the 'memorized' information available enables the RNN model to not simply map the input sequence to output directly, but also to make use of the previous inputs for predicting the next outputs. In this way, the use of an RNN allows relevant contextual information to be used for classification. Previous studies have highlighted its effectiveness in time series prediction, such as speech recognition, text classification, and music transcription [27, 4, 65].

However, the range of contextual information that the standard RNN can process is quite limited. One of the problems during sequence information processing for an RNN is that the influence of previous inputs on the current hidden layer can either decay or blow up. The problem of *vanishing gradient* [32, 33] makes it difficult for a simple RNN to capture information which requires more than 10 time steps between the input and the target prediction [32]. In contrast, the architecture of LSTM can overcome the problem of vanishing gradients by applying memory cells in the hidden state instead of non-linear hidden units operated by RNN. As explained in Section 3.1.2, using functional memory cells allows the LSTM to store and access information over long periods of time, thus avoiding the problem of vanishing gradients. Previous research has proven the success of LSTM's in tasks that require long-range memory for processing such as speech recognition [27], sentiment analysis of sentences [70], and context free and context sensitive grammars [26]. Based on the previous researchers, the success of LSTM's in modeling long-range information also leads us to explore its effectiveness in role-filler prediction.

Previous applications indicate the success of LSTM's in processing global contextual information for sequence inputs. However, LSTM's do not have strong attention ability for each hidden state of the units within the global context [41]. Due to the recurrent structure of LSTM's and RNN's, the contextual information (hidden representation) of each step is fed to the next one. Therefore, for each time step, the current contextual information is only influenced by the previous steps and is quite local [41]. In contrast, as mentioned in Section 3.1.3, an attention mechanism could help to capture additional features in the sequential input. The use of an attention layer in the NN can serve to indicate which segments of the sequence

are more important or relevant than others for specific tasks. Recently, attention mechanisms were shown to be successful in many NLP tasks such as machine translation [42], caption generation based on images [75] and capturing 3D action [41]. To our knowledge, there are no previous studies focused on attention learning for thematic role-filler predictions or selectional preferences based on attention on the sentence level. Consequently, it is necessary to explore its performances in these types of tasks.

3.2 Factorized tensor for word-role embedding and classifier

As mentioned in Chapter 2, the specific thematic role of a given word depends heavily on its contextual information. In the example *boy eats cake* and take care of the boy, the thematic role for the *boy* differs, as it is *agent* in the first case and *patient* in the latter case. This is the same for target role-filler classification, in which providing different contextual information on whether boy is the *agent* or *patient* changes the filler verb filler. In that case, how to solve the problem of role and word information sharing is also one of the challenges of our project.

The naive way of solving this problem is to train separate models with the word having different roles as input. However, given the raw number of n in our task, the number of models needing modification would be n^2, which is expensive and time consuming to train. It is also difficult to preserve interaction between different models, as target role-filler prediction requires several roles to be provided as information.

As suggested by (Tilk et al. [71]), to solve this problem, we will adapt a role-specific word embedding matrix and classifier matrix across the model. These two shared matrices enable us to combine all the input words together with their role into a single model. The advantage of having a single model for training is that we avoid needing to have multiple models for training. This also enables the interaction between input word-role pairs.

Furthermore, as mentioned in the previous chapter, instead of using a one-hot vector as input, we will use a word embedding for training. The use of word embeddings is due to the fact that words are represented as vectors, so those belonging to similar semantic classes can be mapped to nearby points. The same idea is also applied to role embedding. There are several possible methods for word-role pair embedding, such as addition, concatenation, and multiplication. However, simple addition can not preserve the relation between word-role pairs, as discussed in Chapter 2. While concatenation is common for word vector composition in NLP tasks, with respect to word-role pair composition, we may need to add another layer such as *kernel* in the convolutional NN for further feature extraction. After comparison, we will follow the method of element-wised multiplication for word-role pair vectors, as its efficiency has been proven for word-role embedding [71]. To solve the problem of parameter

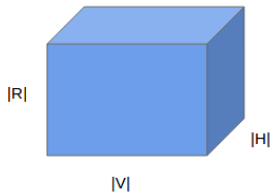


Fig. 3.5 Three-way tensor of size $|V| \times |R| \times |H|$

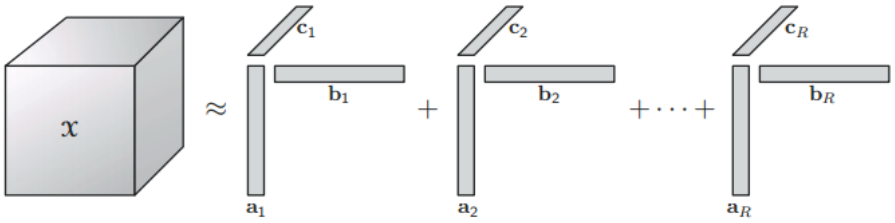


Fig. 3.6 CP composition for an order three tensor

sharing between the word-role embedding and classifier in the model, a three-way tensor is formed as shown Figure 3.5 below.

Even though the three-way tensor enables parameter sharing between the word-role matrix and classifier matrix, given the vocabulary size $|V|$ of 50,000, role number $|R|$ of 7, and vector size as $|H|$ of 256, the number of parameters for model learning will be around 85M. To reduce the number of parameters in the three-way tensor, we follow a **Tensor Decompositions** proposed by (Hitchcock [31]). According to [31], a tensor can expressed as the sum of a finite number of rank-one tensors. This idea is further improved by (Carroll and Chang [12]), called the CANDECOMP/PARAFAC decomposition (CP). CP decomposition factorizes a tensor into a sum of component rank-one tensors. Given a three-way tensor in our case, its decomposition can be calculated as [37]:

$$\chi \approx \sum_{r=1}^{R} a_r \circ b_r \circ c_r \tag{3.9}$$

where R is a positive integer and a_r and $a_r \in R^I, b_r \in R^J, c_r \in R^K$ for $r = 1, ..., R$. Element-wise, it can be written as :

$$x_{ijk} = a_{ir} b_{jr} c_{kr} \tag{3.10}$$

where $i = 1, ..., I, j = 1, ..., J, k = 1, ...K$. This is illustrated in Figure 3.6 [37]

So assuming the lateral slice of χ represents the role-specific weight matrices, where j refers to the specific role, we can then write each matrix as [71]:

$$W = Adiag(rB)C \tag{3.11}$$

where r is the role vector and $diag$ is the diagonal matrix with the argument vector on the main diagonal and zeros elsewhere [71]. A and C represent the factor matrices into which the embedding tensor is factored. Thus for each input word, its role-specific word embedding vector e is computed by taking from the factored embedding tensor the row corresponding to the word being indexed and the column corresponding to the role being indexed, as we explained above:

$$e = wA_e diag(rB_e)C_e \tag{3.12}$$

where w, r refer to the *word* and *role*. A similar method is applied to the factored target role specific classifier tensor: the classifier is computed by Equation 3.13, given the context information, which is the hidden state h from the NN, the target role t_r.

$$\begin{aligned} c &= hA_c diag(t_r B_c)C_c \\ y &= softmax(c+b) \end{aligned} \tag{3.13}$$

The prediction for target role-filler is then calculated with $softmax$, the output of which represents the probability distribution over the output vocabulary.

3.3 Model Architecture

Based on the proposed method in 3.1, 3.2, the diagram for general structure of the models adapted in our project is as follows:

Input: As sequence of words i_w; The set thematic roles corresponding to each word i_r; The target role we need to predict t_r;

Output: The word as filler for the target role t_w;

1. Set the required input (i_w, i_r, t_r) into NN model
2. With information of i_w, i_r, get the role-specific word embedding e from Equation 3.12
3. Get the hidden states h_r of embedded vectors with Equation 3.1, Equation 3.4, or Equation 3.5
4. (Optional) Get the weighted context information c with Equation 3.6 and 3.7 by attention mechanism
5. With the input information of t_r and c (or h_r), get the target-role specific classifier in Equation 3.13
6. Predict the possible role-filler t_w for target role with $softmax$ in Equation 3.13

return t_w;

Chapter 4

Experimental Setup

In this section, we will cover detailed information on the training data and its preprocessing to meet the requirements for our project. Additionally, we will also cover the preprocessing for Word2vec, as it is an important representation of word meanings in this Neural network-based approach. Then, we will list several types of NN models we have tried previously for thematic role-filler prediction and the possible evaluation methods for model performance. At the end of this chapter, we will also provide the technical details for NN training and optimization.

4.1 Data sets

The training data used in our project is from UKWaC, which is constructed from web pages crawled from the *.uk* web domain, using medium-frequency words from the BNC British National Corpus) as seeds. The corpus consists of 138 million sentences with about 2 billion words.

Corpus Annotation To annotate the sentences in UKWaC, SENNA is used in the project. SENNA is a high performance role labeler and generates PropBank style role labels[20]. The advantage of using SENNA as a Semantic Role Labeling (SRL) tool is that its data processing performance is fast and robust and could also process large amounts noisy data such as in UKWaC [62]. Its annotations are based on raw text, and for SRL it does not require additional information such as syntactic parsing trees. The semantic roles it annotates includes ARG0 and ARG1, which are verb-specific roles. According to PropBank [10], most of the Arg0 label is assigned to arguments which are understood as *agents, experiencers*. The ARG1 label is usually assigned to the *patient* argument while other modifiers such as location (LOC) and manner (MNR) are also provided by SENNA.

Head extraction After SRL, the heads for each semantic role are also extracted according to the work described by Sayeed et al. [62]. Two types of head-finding algorithm are

```
<s>
   <rawsent>but her selected information has a welcome sharpness :</rawsent>
      <governor>have/vbz/5</governor>
      <dependencies>
         <dep algorithm="malt" source="but/cc/1" text="but" type="am-dis">but/
cc/1</dep>
         <dep algorithm="malt" source="her/prp$/2 select/vbn/3 information/nn/4"
text="her select information" type="a0">information/nn/4</dep>
         <dep source="have/vbz/5" text="have" type="v">have/vbz/5</dep>
         <dep algorithm="malt" source="a/dt/6 welcome/jj/7 sharpness/nn/8" text="a
welcome sharpness" type="a1">sharpness/nn/8</dep>
      </dependencies>
   </predicate>
</s>
<s>
```

Fig. 4.1 Sample data in training corpus

introduced. One is based on the syntactic dependency information produced by the parser *Malt*. The other *linear* method is based on the Part-Of-Speech tagger information which is provided by SENNA. This is relevant because during training processing, these heads will be used as the target word for the specific role-filler. According to previous work by Sayeed et al. [62], all of these information is stored in XML structured files: As shown in the figure above, for each sentence, the *governor* (verb predicate) , and its *dependencies* is stored. While for each dependency, the *source information, semantic role, head* are save. In addition, all words in *source* are replaced by their lemma, together with the POS information and their position in the sentence.

Predicate-arguments Pair Extraction During data processing, each word is extract from the sentence according to their semantic roles and words positions in the sentence. Furthermore, their partof speech tagger (POS) information is also extracted. Lastly, we find the top 50,000 most frequently used words and map all the corresponding words, POS, and semantic roles as integers. The final format for each sentence is stored as a *list* in python, which consists of elements in thematic roles. Each thematic role element contains information on [Role, Head,Head$_P$OS,Noun phrase, Noun phrase POS] (POS refers to Part-of-Speech tag). An example input and its explanation is listed as below.

$$[[1,304,17,[1346,50001,304],[29,42,17]],[0,1088,33],[2,783,12,[50001,498,783],[42,5,12]]]$$

$$[[Role_1,Head_1,Head_1 \quad POS,[Noun \quad phrase_1],[Noun \quad phrase_1 \quad pos]],[Role_2,Head_2,Head_2 \quad pos]$$

$$[Role_3,Head_3,Head_3 \quad POS,[Noun \quad phrase_3],[Noun \quad phrase_3 \quad pos]]$$

Data Segmentation After extraction, around 200 million verb-argument samples are extracted from the corpus. We take 10% of these as validation data and another 10% as testing data. The rest is used as training data. During the training process, for sentences with multiple arguments, we assign each argument as the target role for each training sample.

4.2 Word embedding with Word2vec

As discussed from previous chapters, word embeddings have been popular for different NLP tasks, including tasks for thematic fit and word preferential selection. In NLP tasks, word embedding refers to representing words using vectors. These vectors are created by mapping other representation of words, such as integers or one-hot encoding vectors to a low dimension vector. Following such previous research, we will also apply word embeddings to model training.

In brief, the advantages of using a word embedding instead of a simple one-hot encoding lie in two aspects: firstly, with the increase in vocabulary size, the dimension of the one-hot vector for each input word may become extremely large. As a result, the weights in the weight matrix from the connected hidden layer would also increase. In contrast, the dimension of a word embedding is independent of the vocabulary size or number of input items and could be kept in a lower dimension. Secondly, one-hot encoding cannot provide any semantic information for input words. However, word embeddings can present semantic information about input words when trained effectively. This feature of word embeddings may benefit the model's prediction performance.

Word2vec, introduced by (Mikolov et al. [50]) allows words to be mapped to vector representations using neural networks. This method enables us to represent words with low dimension vectors in many NLP tasks. Consequently, we will also employ this method for word embedding. There are several variations for the application of word2vec. However, the general idea is to capture the words' semantic meaning by checking which words are possible to co-occur within certain window size. It learns the words' semantic meaning by learning their contextual information where the words are used in the corpus.

In the application of word2vec, our first concerns is to use the pre-trained word2vec from by Google word2vec model. For the Google word2vec model, the word embeddings are trained on the Google News dataset, with a vocabulary of 3 million words. One of the essential advantages of using pre-trained word2vec is that, it can reduce the number of training parameters in the model (compared with training word2vec within the model architecture). And it will decrease the training time. However , one drawback of using Google word2vec specifically in our project is that, all the word representations are derived from American English words (trained on the corpus of Google news). In order to load word2vec model offered by Google, we need to firstly convert all the words in Birth English into American English (such as *biscuit VS cookie* or *flat VS apartment* or *colour VS color*). During conversion, it's shown that, among the top 50,000 words in our training corpus, 16,167 words are missing of vector representation, either due to the words gap between two corpora or the failure of conversion from British English to American English. Some of the missing words and their frequencies in the training corpus are listed in the Table 4.1.

Missing word	Frequency	Missing word	Frequency
and	881589	enquiry	69843
prise	60509	judgement	34425
authorisation	10371	acknowledgement	8918
londoner	7878	livingstone	6643

Table 4.1 Some important missing words and their frequency during British-American English conversion

Consequently, the word embedding is finally trained based on our training corpus, UKWac, with 2 million vocabulary. There are several training algorithms for word2vec and the one used for our word2vec model is Continuous Bag of Words (CBOW) Learning with hierarchical softmax [48]. The learning of word vector is based on the relationships between pair of words. According to Mikolov et al. [48], the CBOW model predicts a word given its context. Here context refers to both the right and left contexts words, and the word order is ignored.

In order to feed our data properly into the training model of word2vec, we extract all raw sentences from the XML described in Section 4.1 . Later, we use the python package *nltk* and the lemma dictionary produced previously from SENNA to normalize all the sentences and then convert each sentence into a list of words. During the training of word2vec, we use a context size of 5 and exclude words with frequencies lower than 10 in our corpus. Finally, we produce word2vecs with 200 dimensions, 300 dimensions and 400 dimensions separately to see its effect on word meaning representation during the model training.

Apart from the word embedding, we also apply the POS information and role information for model training. Although embeddings are mainly applied for word vectors, they are not exclusive applications for NLP tasks. Regarding the sequence of POS taggers or roles as integers, the word2vec mechanism can also transfer them into embedding vectors. With this application, we can capture the meaning representation for these role and POS taggers as well.

4.3 Experimental models

In Chapter3 and Chapter 2, we have discussed several possible methods for word-role embedding and also the context information learning. With this in mind, during the experiment period, we tried several NN architectures, each of which we shall introduce in the following section.

We believe that in addition to the word embedding, the POS for each word will also help to learn the context for meaning extraction. Of all words in the noun phrase of an argument,

the head noun and its adjective modifier or noun modifier will usually influence the role-filler prediction. So, word $w_1, ..., w_i$ in each input sequence is represented as the concatenation of the word embedding and the POS embedding. Research by Tilk et al. [71] has already proven the efficiency of factored role-specific word embedding and factored classifier for parameter sharing among all words in a vocabulary with different role information within a given context. Because of that, we will generally adapt this method as described in Section 3.2, representing each word in the input sequence as its role-specific embedding vector e. Where not explicitly noted, we regard the input for each model as the sequence of factored role-specific word embedding e_i from Equation 3.12 instead of w_i word vectors.

During the experiments, in general, we focused on two types of contextual information extraction and composition. The first method focuses only on the noun phrase information presented in the *Agent* and *Patient* fillers. The second type takes the whole sentence as input for information representation learning.

4.3.1 Sentence based NN model

As discussed previously, RNN models, especially LSTM models, can be used to deal with a sequence of input. With its functional gates, a LSTM can capture important information from the previous state as well as remember information from current state. These features enable LSTM's to deal with sequences from long-dependencies. The attention mechanism can interpret the model by showing the importance of the elements for the input sequence for prediction. With the attention model, we can create a context vector and attend to different weights on the elements, avoiding the necessity of encoding all the information only in the last hidden state. Given this, it is necessary to explore the effect of LSTM and the attention mechanism in our project. Bearing this in mind, we have explored different types of models.

Non-incremental model based on word-role embedding

The method used for non-incremental model (NON_WR model) is described in [71]. After we get the role-specific word embedding from Equation3.12, we simply sum all the embedding elements e_i together to get the sentence representation. We then apply a parametric rectifier non-linearly to get the hidden state representation [30].

$$h = PReLU(\sum_{i=1}^{N} e_i + b) \tag{4.1}$$

where h will be applied to the role-specific classifier c . The target word is then predicted based on Equation 3.13. We set this model as a baseline for meaning composition.

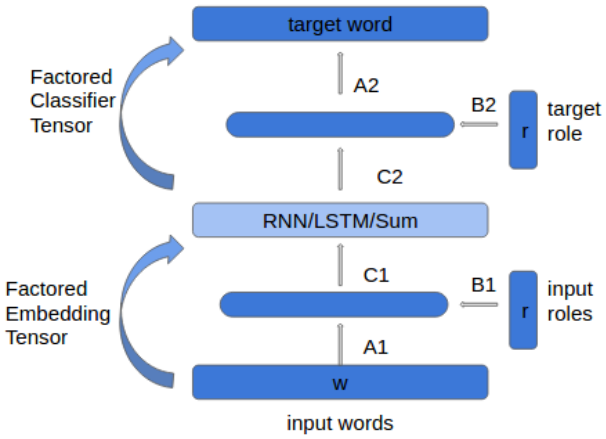


Fig. 4.2 General structure for first three models

Simple RNN model based on word_role embedding

The difference between the simple RNN model (RNN_WR) and NON_WR model in the previous section is that instead of summing all the elements from the word-role embedding, we apply a simple RNN layer to the embedding layer. Then the hidden state h_i is captured by the previous hidden state h_{i-1} as well as the current embedding e_i. Detailed information is shown below:

$$h_t = PReLU(We_t + Uh_{t-1} + b) \tag{4.2}$$

where W, U are weight matrices in the RNN layer. This RNN layer adds information about the previous hidden state to the current embedding through W, U. The last hidden state h_n is then used for role-specific classifier embedding and target word prediction, again by Equation 3.13.

LSTM model based on word_role embedding

Similar to the the RNN_WR model, the LSTM-based word_role embedding model (LSTM_WR) takes the embedding layer as input, then returns a single hidden state h for the input sequence, with Equation 3.4. Then like the RNN_WR, the hidden state is used for classifier embedding and target word prediction with Equation 3.13. The general structures of NON_WR, RNN_WR, and LSTM_WR models are illustrated in Figure 4.2

Attention based on RNN layer

Recently, attention mechanism added on top of RNN layer has been applied in several NLP tasks [80, 42]. We will also follow their application and set weighted scores α_i for each step

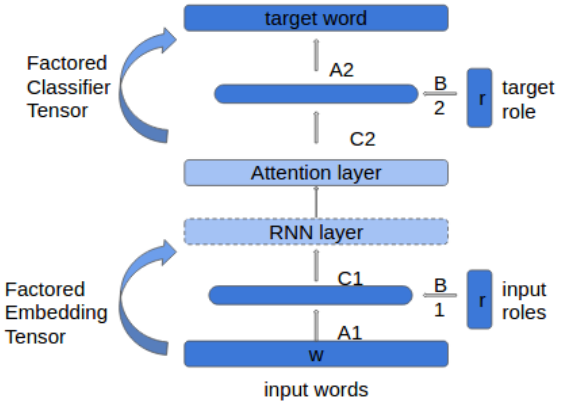


Fig. 4.3 General structure for attention based models

of hidden state h_i in the RNN layer (usually LSTM or GRU). The context vector c is then captured based on the hidden state h_i and the attention score α_i. Detailed information is mentioned in Section 3.1.3. Instead of using h for role-specific classifier embedding mention in LSTM_WR model, we will use the c in Equation 3.13 for target word prediction.

Attention based on word_role embedding

Most of the attention mechanisms are applied to the RNN layer, but Choi et al. [15] introduce their approach of directly extracting attention score based on the word-embedding. Besides, the recent state of the art for MT [74] also indicates the potential of learning attention simply based on the word embedding. Inspired by their work, we propose to create the context vector by only focusing attention on the word_role embedding instead of hidden states from the RNN or word embedding (Attention_WR). In that case, given the word_role embedding e_i as input sequence, the attention score α and context vector c are calculated as :

$$\begin{aligned} a_t &= tanh(We_t + b) \\ \alpha &= softmax(a_1, ...a_t) \\ c &= \sum_{i}^{T} \alpha_i e_i \end{aligned} \tag{4.3}$$

The attention vector is then applied in Equation 3.13 again for target word prediction. The general structure for the attention based model is shown in Figure 4.3.

Attention based LSTM with Word embedding only

Even though the factored role-specific word embedding and role-specific classifier embedding have been proven to be effective in parameter sharing and thematic role-filler prediction, we also experimented with a model of Attention_LSTM_W2V. This model is similar to the Attention_LSTM model. However, instead of taking a word-role embedding as input for the LSTM layer, the Attention_LSTM_W2V model only takes information from the word embedding. All the other computation procedures are the same as the Attention_LSTM model as mention above. We also set this model as a baseline to check the performance of rolespecific word embedding.

4.3.2 Noun phrase based attention model

In noun phrase based model (NP_attention model), we only extract noun phrase information from the *Agent* and *Patient* roles and then use a shared layer to learn the weighted information of each NP. After obtaining the weighted context information c_i for each NP, we sum them together with the word-role embedding from other roles. Then the context information for the whole sentence is used for role-specific classifier embedding to predict the target role-filler. The specific learning diagram for this model is shown as :

---

for argument in *agent, patient* **do**
    extract NP information;
    for w_i in NP , get the role-specific word embedding e_i
    Get the recurrent hidden state h_i for each embedding vector e_i.
    Get weighted NP representation C_np with the attention mechanism
end for
for argument in other roles **do**
    extract the head of argument
    get role-specific word embedding e_i of head
end for;
sum the new representation of each role together to get the new hidden state h_{c1}
add a non-linear activation function on h_{c1} to get new hidden h_{c2}
get role-specific classifier c based on h_{c2} and target role t_r
predict target word with the classifier c

Table 4.2 shows the general information of the models we've experimented during the project.

Model name	Description
NP_Attention	Attention mechanism applied on LSTM layer within NP for *agent* and *patient*
NON_WR	Nonincremental model applied to the wordrole embedding
RNN_WR	Simple RNN model applied to the wordrole embedding
LSTM_WR	LSTM model applied to the wordrole embedding
Attention_LSTM	Attention mechanism applied on top of LSTM layer which is based on wordrole embedding
Attention_WR	Attention mechanism applied on top of wordrole embedding
Attention_LSTM_W2V	Attention mechanism applied on top of LSTM layer which is based on word embedding

Table 4.2 Models and description

4.4 Evaluation Matrices

4.4.1 Perplexity

Perplexity (PP) is one of the most frequently used evaluation method for language models, as it can be easily calculated and its score indicates the probability distribution of a given model. Given a language model and a test corpus, the perplexity of language model (LL) can be calculated as [35]:

$$PP = exp(-\frac{1}{N}\sum_{i}^{N} log(p(w_i|h_i))) \tag{4.4}$$

where N is the number of words in the test corpus, w_i is the the word to be tested, and h_i is the history of w_i. The idea underlying PP is that, a lower score of PP indicates better performance of the model to predict words in the test corpus.

During application, instead of testing the probability of each word in the test data, we will only test that of the target role-filler. For example, given the input sequence below, if the target word is *medicine*, we will then only count the probability of *medicine*, instead of all other words in the sentence.

- (the sick boy)$_{[agent]}$ takes[verb] ??$_{[patient]}$

During the experiment, it is also noted that PP should be calculated with the same test data set as all models, while the vocabulary size should also be kept the same. The main advantage of PP is that it is fast to perform and independent to other complex systems. Therefore, in this thesis, PP will be the main measurement used for experiments.

4.4.2 Evaluation on thematic fit rating

To check our model's performance and its arguments meaning composition, we also evaluated the effectiveness of our model with human thematic fit rating. Human ratings for thematic fit are usually obtained by asking participants to rate how common or typical the test argument is, given a certain event verb as contextual information [55, 28, 45]. However, most of these data only contain event-argument pairs, such as *secretary address letter* or *speaker address group*, so they rarely contain contextual information. In order to test our model's performance in thematic role-filler with given context information, we built our own evaluation data based on previous studies.

The first type of data we attempted to modify is derived from the experimental stimuli offered by Matsuki et al. [43]. In their stimuli, sentences are marked as *typical* or *atypical* in regards to the filler of *instrument* or *patient* used in a given context. Examples are shown below:

1. (Typical) Jessie used a **shortcut** to avoid the annoying traffic because he heard that there had been an accident on his usual route. He was in a hurry to get to work because he had an important meeting that morning.

2. (Atypical) Jessie used a **payment** to avoid the annoying traffic because he heard that the freeway was jammed. He hated paying tolls, but he was in a hurry to get to work that morning.

3. (Typical) John used a joystick to control the brand-new **game** that he bought yesterday. He had to stop using the regular controller because of the blisters on his both thumbs.

4. (Atypical) John used a joystick to control the brand-new **television** that he bought yesterday. He is a heavy gamer, and likes to control everything with his joystick.

In the first two examples, the judgment of being typical or not is based on the proper use of role filler of *instrument*. Whereas in the second two sentence pairs, the judgment is based on the role filler of *patient*.

Based on the context information given in the stimuli, we rewrite the sentence as:

(3) Jessie avoided the annoying traffic with a **shortcut**.

(4) Jessie avoided the annoying traffic with a **payment**.

(5) With a joystick, he controls the brand-new **game** that he bought yesterday.

(6) With a joystick, he controls the brand-new **television** that he bought yesterday.

There are in total 48 sentence pairs extracted for *patient* and *instrument* role filler prediction. All the samples are annotated with SENNA for their thematic roles and POS tags. Next, the

head noun for each role is manually extracted from the annotated data. After annotation, all necessary thematic roles, words, and POS are mapped into integers as described in Section 4.1.

During evaluation, if the target word such as *shortcut* is an unknown word in our corpus, we skip the sentence pair. The evaluation method follows that of Lenci [40], Tilk et al. [71], which is based on *accuracy* of expected direction of thematic fit difference . Given the modified sentences in Example 1 and Example 2 as the query *Jessie avoided the annoying traffic*, and given *instrument* as the target role, if the model assigns a higher probability to the typical item *shortcut* than to the atypical one *payment*, we count it as an accurate prediction. The same method is applied to the evaluation of *agent* roll filler.

The second type of evaluation data follows the experiment stimuli offer by Amsel et al. , in which they conducted research on Event knowledge activation during language processing as recorded with an electroencephalogram (EEG). The stimuli they used offer seven types of modality information. Given the first sentence as contextual information, the experiment participants are tested on their reaction to four word fillers for the second sentence. Examples of this data are shown below.

(7) Jordana's husband slipped a special anniversary gift around her wrist right before the big dinner. All evening the guests were staring at her sparkly ??? in admiration. [bracelet, sandals, guests, pony]

(8) Yesterday, a baby cow was born on the floor in my family's old barn. It lay on the soft golden ??? while it rested. [hay, bus, farm, soldier]

Four words are provided to the participants within the given context and participants are asked to fill in the blanks for the second given sentence of each sentence pair. In the first example, among the four words provided, **bracelet** is the *expected word*. The second one **sandals** is the socalled *Perceptuomotorrelated* word. The third word **guests** is the *event-related* word while the last one **pony** is the *unexpected word*. Given the example above, we only adopt the *expected word* and *unexpected word* for the purpose of our evaluation. As we need to modify the given stimuli to fit out purpose of evaluation, without enough contextual information, the *perceptuomotorrelated* word and *event_related* word usually are not atypical enough for the model to distinguish the contextual information. The modified sentences are shown below:

(9) All guests admire the sparkly [bracelet, pony].

(10) The baby cow lay on the soft golden [hay, soldier].

The evaluation method for this is similar to the previous one from Matsuki et al. [43]'s stimuli. Given the query of *the sparkly bracelet is admired by* and *the sparkly pony is admired by*, we proceed to analyze the probability of the *agent* filler *guest* in each model's prediction. If the

probability of *guest* in the first query is higher than that in the second, we record a success. Later, accuracy is also compared amongst the 100 sentence pairs.

4.5 NN training

4.5.1 Implementation

We implemented the models mentioned above with the Deep learning framework Keras, and the pre-trained word2vec is based on the python package Gensim. During processing of the evaluation data, we also use SENNA for SRL and POS tagging.

During the training of the NN, we also adapted several methods for modeling optimization. To prevent over-fitting, we used dropout and early stopping. In this context, dropout refers to dropping the hidden state units out of the NN randomly during training [68]. In effect, this removes the hidden state and all its incoming and outgoing connections. Another method to prevent over-fitting is early-stopping and reducing the learning rate, which is monitored on the validation loss. As mentioned before, during data segmentation we select a separate set of data as validation data. After each epoch of training, the loss for the validation data is measured. If the loss does not decrease, we reduce the learning rate by a certain proportion. However, if the learning rate reaches a threshold and performance does not improve, training is halted automatically. In order to tackle the problem of gradient explosion, we have also used the gradient clipping method. The norm of the gradient is clipped if its value is greater than 1. During training, we tried different variants of LSTM. The results indicate that GRU performs better than traditional LSTM. Therefore, the result reported later is for a GRU-based LSTM model. As for the optimizer, we have used AdaGrad, following the suggestion of Tilk et al. [71]. The advantage of AdaGrad is that it is able to increase the learning rate for more sparse parameters while decreasing the learning rate for those that are less sparse. For the GRU version, we also attempted the Adam optimizer. Its performance was not worse than AdaGrad. Therefore, we may also explore the adaptation of Adam in all the other models in future work.

In conclusion, to find the correct hyperparameters for training, we have tried different values for learning rate, hidden layer, dropout rate etc. However, due to the long computational time required for training during each experiment, we did not perform all the possible combinations of the parameters. Because of that, it is possible that there are more accurate parameters than those used in this project.

Model name	LR	H	E	N	dropout	Optimizer
NP_Attention	0.01	256	256	$5*10^7$	0.3 / 0.4	AdaGrad
Non_WR	0.005	256	256	$5*10^7$	0.3 / 0.4	AdaGrad
RNN_WR	0.01	256	256	$5*10^7$	0.3 / 0.4	AdaGrad
LSTM_WR	0.001	256	256	$5*10^7$	0.3 / 0.4	Adam
Attention_LSTM	0.005	256	256	$5*10^7$	0.3 / 0.4	AdaGrad
Attention_WR	0.005	356	356	$5*10^7$	0.3 / 0.4	AdaGrad
Attention_LSTM_W2V	0.001	256	256	$5*10^7$	0.3 / 0.4	AdaGrad

Table 4.3 Number of hyperparameters applied for each model

4.5.2 Hyper parameters

Table 4.3 shows the training parameters for each model. **LR** refers to learning rate; **H, E** to hidden layer size and embedding layer size, and **N** to the number of training samples per epoch.

Chapter 5

Results and Discussion

In this part, we will present the results for different models' with respect to their perplexity and performance in thematic fit evaluation.

5.1 Perplexity

Table 5.1 shows the overall results of the perplexity we have tested on all the models (the lower perplexity the better performance). In the table, *PP* stands for the perplexity score, *Val_acc* refers to the validation accuracy. while *Test_acc* indicates the accuracy of test data. Apart from the models that we have described in the previous chapter, we also tested the perplexity on (Tilk et al. [71]) model of Head_Noun_RNN. Instead of taking the whole sentence information into consideration, Head_Noun_RNN only takes the content word of each thematic role as the input word, and it applies a simple RNN layer after the rolespecific word embedding. Later, Head_Noun_RNN used the last hidden state of RNN as the input for

Model name	PP	D_PP	Val_acc	Test_acc
NP_Attention	**887**	3.41	**0.063**	**0.063**
Non_WR	985	3.45	0.055	0.055
RNN_WR	1088	4.32	0.055	0.055
LSTM_WR	**834**	3.28	**0.066**	**0.065**
Attention_LSTM	1012	3.82	**0.064**	**0.065**
Attention_WR	1374	5.00	0.042	0.043
Attention_LSTM_W2V	5838	13.23	0.020	0.021
Head_Noun_RNN	913.3	3.38	0.062	0.062

Table 5.1 Perplexity and accuracy comparison between different models

the factored classifier embedding in Equation 3.13. The comparisons between our models and Head_Noun_RNN can illustrate whether our model can perform successfully for content information extraction of each argument, when processing the sentence or phrase.

Comparison with baseline Attention_LSTM_W2V model

As described in Section 4.3, we use the model Attention_LSTM_W2V as a baseline for testing the performance of word_role embedding in our project. Different from all the other models, Attention_LSTM_W2V leans the context information simply based on the word2vec embedding instead of the factored role_word embedding. In general, the PPL for all these models are quite different. But the results show that, Attention_LSTM_W2V has the highest PPL score and the lowest accuracy for either validation data or testing data among all these models. Similar to the results from (Tilk et al. [71]), this proves the necessity of using factored role_word embedding for thematic role-filler prediction tasks.

Comparison with baseline Non_WR model

In Section 4.3, we also introduce another baseline model Non_WR, where we apply addition for the word_role embedded vectors in the input sentence. Firstly, out observation found that Non_WR and RNN_WR have similar performance regarding to perplexity or validation and test accuracy. Whereas in (Tilk et al. [71])'s paper, they also reported the similar performance of nonincremental model and incremental model for thematic role-filler prediction based on the content noun. The incremental model in [71] is similar to our model of RNN_WR. Secondly, LSTM_WR out-performs both RNN_WR and Non_WR in PPL, which again indicates the advantages of LSTM in processing long sequence of data with its functional memory cells. Thirdly, the PPL performance of Attention_WR (where context information is represented by the weighted addition of word_role vectors) is not as ideal as the Non_WR model. This may indicate that for our task, the attention scores, which are learned based on the global contextual information among the whole sentence, might not be as effective as that of the single local information (the vector information for each input of word_role embedding or the hidden state of RNN).

Comparison among attention mechanism

During the experiment, we applied four types of attentional learning for sequence information processing. Among these models, NP_Attention focuses on the noun phrase meaning composition whereas the other three models apply an attention mechanism for the whole sentence. Table 5.1 shows that all the attention mechanisms applied in sentence level have a higher perplexity than that from the noun phrase level. Moreover, the attention applied in noun phrase level outperforms that of the single Head_Noun_RNN. This result on one hand

indicates the effectiveness of NP_Attention in contextual information extraction. On the other hand, it also proves the power of LSTM as hidden state learning than that of a simple RNN, because in NP_Attention, the weighted contextual information is based on the LSTM hidden states.

In summary, the result shows that, LSTM_WR has the lowest perplexity score among all the models. It's performance is much better than the simple RNN model or the non-incremental model. This again proves the effectiveness of LSTM in dealing with long sentence. While, for the attention models, all the attention mechanisms applied in sentence level have a higher perplexity than that of the single RNN or LSTM. However, as we observe from the result, the NP_attention model show similar performance to the LSTM_WR, which may indicate that the attention mechanism is more effective in dealing with noun phrases (shorter sequence of words) rather than long sentence. Furthermore, both the LSTM_WR and NP_Attention has better performance than the model in (Tilk et al. [71]), which may indicate that, LSTM_WR and NP_attention can extract the content word information among the sentences or phrases. Meanwhile, Attention_LSTM_W2V has the highest perplexity of all these models , this result proves the effectiveness of the rolespecificword embedding for parameters sharing in the model training process.

We also provide the accuracy results for validation and test data. Compared with all the other models, LSTM_Attention and LSTM_WR has a better performance. Despite the higher perplexity score in testing data, we are expecting LSTM_Attention to have better performance for thematic fit evaluation based on its high validation accuracy. Meanwhile the model with word2vec instead of word-role embedding as input has the lowest accuracy. Likewise, the result shows the necessity of using factored role_word embedding for thematic role-filler prediction.

5.2 Thematic fit evaluation

During the experiment, we have evaluated thematic fit on two types of data. The first type of data is based on the experiment stimuli mentioned in (Matsuki et al. [43]), where we focus on the prediction of *patient* and *instrument*. The second type of evaluation data is modified from the paper of [3] for *verb* and *agent* filler prediction.

5.2.1 Thematic fit difference on *Patient, Instrument* fillers

As shown in Chapter 4, the example sentence pairs for *patient* and *instrument* fit evaluation is listed below:

(11) Jamoes caught the elusive baseball with a *glove* before it fell into someone else's hands

Model name	Patient	Instrument
NP_Attention	0.558	**0.617**
Non_WR	**0.574**	0.575
RNN_WR	0.460	0.550
LSTM_WR	**0.574**	**0.625**
Attention_LSTM	0.510	0.525
Attention_WR	0.489	0.550
Attention_LSTM_W2V	0.510	0.525
Head_Noun_RNN	0.553	0.575

Table 5.2 Evaluation on thematic fit difference for *Patient* and *Instrument*

(12) Jamoes caught the elusive baseball with a *net* before it fell into someone else's hands

Given the same context, if the model can assign a higher probability to the typical instrument filler *glove* than an atypical one *net*, we count the model's prediction as an accurate one. During evaluation, we will skip the sentence pair if the target word is an unknown word in our vocabulary. Finally, there are 47 sentence pairs for *patient* filler and 40 sentence pairs for *instrument* filler evaluation. Table 5.2 shows the accuracy result based on the evaluation data. Unfortunately, there is no significant difference among all the models, as the number of sentence pairs for evaluation is limited. However, among all the models, the NP_Attention has a higher accuracy on instrument prediction while LSTM_WR has a slightly better result for *patient* role prediction.

5.2.2 Thematic fit difference on *Agent, Verb* fillers

In this section, we evaluate all of the models' performance on *Agent, Verb* fillers' prediction based on stimuli in (Amsel et al. [3]). The example data is listed below:

(13) The **artists choose** a bright red *lipstick* in the makeup show.

(14) The **artists choose** a bright red *concrete* in the makeup show.

In the example above, *a bright red lipstick* is an typical example in the given context whereas *a bright red concrete* is an atypical one [3]. Similar to the evaluation in Section 5.2.1, given two different context and same role-filler (here *agent, verb*) as the target word, if the model can assign a higher probability to the word in the typical sentence than that in the atypical sentence, we count model's prediction as an accurate one. This evaluation can help to illustrate the models' performance in context information extraction and their ability to distinguish typical and atypical context. During evaluation, if the target word in the sentence is out of vocabulary, we will skip that sentence pair. In total, there are 115 sentence pairs

Model name	Verb	Agent
NP_Attention	0.757	0.739
Non_WR	0.686	**0.748**
RNN_WR	0.652	0.730
LSTM_WR	0.765	0.739
Attention_LSTM	**0.800**	**0.748**
Attention_WR	0.626	0.613
Attention_LSTM_W2V	0.130	0.172
Head_Noun_RNN	0.783	0.703

Table 5.3 Evaluation on thematic fit difference for *Verb* and *Agent*

available for *Verb* filler prediction and 111 for *Agent* filler prediction. The result for thematic fit difference evaluation is illustrated in Table 5.3.

From Table 5.3, we observe that, the baseline model Attention_LSTM_W2V has the lowest accuracy in both *verb* and *agent* filler prediction. This again indicates the necessity of word_role pair embedding for our task. For another baseline model Non_WR, it has highest accuracy in *agent* filler prediction, together with the Attention_LSTM model. Whereas NP_Attention, LSTM_WR and Attention_LSTM outperform the baseline for *verb* filler prediction. Despite the high perplexity, Attention_LSTM achieves the best performance for either *verb* or *agent* filler prediction. It may indicate the effectiveness of Attention_LSTM in real data prediction and its penitential in processing whole sentence as input sequence .

As mentioned in Section 5.1, we also test the performance of content word prediction model Head_Noun_RNN. The evaluation result shows that, for the prediction of *verb* filler, only Attention_LSTM model outperform the one based on content words. This on one hand proves the effectiveness of Attention_LSTM in contextual information extraction. On the other hand, it may indicate that during *verb* filler prediction, the contextual information is not as important as that for *patient* fillers. Whereas with the aspect to *agent* filler, most of the models we've experimented outperform the single head model with only content word. This also proves that the content of *agent* fillers depend heavily on their contextual information. Furthermore, both the Non_WR and Attention_WR models have highest accuracy for *agent* filler prediction.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this work, we explored methods to predict thematic role-fillers based on phrase or sentence information. To fulfill the task, we trained our words embedding with CBOW algorithm for word meaning representation, as it showed an advantage in presenting semantic similarities as well as keeping vector dimension fixed despite the change of vocabulary size. For the overall task, we proposed two methods: first of all, to process sequence information, we propose the use of RNN for sentence or phrase meaning learning. To further explore the weighted meaning composition in sequence information, we explored the use of attention mechanism on top of either RNN or word embedding. Second, to learn the shared parameters between word and role pairs, we followed the work from (Tilk et al. [71]) to build factored role-specific word embedding matrices for word meaning embedding and later to build factored role-specific context embedding matrices for target word classification.

The performance of models is later evaluated based on perplexity and thematic fit related to human rating. The perplexity result shows that the attention mechanism is more effective in phrase level meaning composition than sentence level. One possible reason is, in phrase level, the head noun will play a dominant role in thematic meaning representation, thus it was easier for the model to assign the higher score for it to learn. Meanwhile the LSTM neural network proved its power in processing long dependency sentences. Furthermore, our results prove the effectiveness of using wordrole embedding matrices for parameter sharing during the role-filler prediction task. All the models adopting a role specific word embedding outperform the ones using only word2vec as input. The use of non-incremental model and simple RNN model shows similar performances in our task, which are similar to Tilk et al. [71]'s experiment results. Whereas, according to the thematic fit evaluation on all the models, the attention model for sentence level prediction shows better performance in both verb and patient filler prediction. This illustrates the importance of attention mechanism in contextual

meaning extraction for our project and also indicate its potential for processing sequence input of whole sentences.

6.2 Future work

6.2.1 Theoretical part

Based on the performance of our task, we observe that, the attention mechanism works better in phrase level compared to the sentence level information processing. How to improve its performance in the sentence level is the further focus of our task. One of the possible ways to deal with the weighted meaning composition in the sentence level is inspired by the paper from (Vaswani et al. [74]). (Vaswani et al. [74]) split sentence into multiple parts which share the identical layer of weight learning. While in every part, the weighted meaning for each word is derived from multihead mechanism. Taking word embedding as input, each word is split into multiple parts (head) from the embedding dimensions. The weighted score is then based on the linear projection of keys and query which is derived from its own embedding feature. As a state of art method in machine translation, it is possible for us to explore its performance in thematic role-filler prediction in the future.

6.2.2 Evaluation

Since we use perplexity for performance comparison, one of the important points we need to bear in mind is to ensure all the models share the same data for testing. Moreover, the test data should be representative. During the experiment, we have applied several processing methods to ensure all the models shared the same data for training and testing while the data is segmented based on the proportion of 80% for training, 10% for testing and the other 10 % for validation. It is possible to explore the effect of cross-validation, by assigning the training, testing, and validation data randomly for each epoch, which would make those data more representative.

Another concern about the evaluation is that, due to the limited number of evaluation data, some performance of the models may not indicate significant differences (for the filler prediction of *patient* and *instrument*). Thus in the future, improvements could be made to collect more proper data for evaluation.

References

[1] Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

[2] Agirre, E. and Martinez, D. (2002). Integrating selectional preferences in wordnet. *arXiv preprint cs/0204027*.

[3] Amsel, B. D., DeLong, K. A., and Kutas, M. (2015). Close, but no garlic: Perceptuomotor and event knowledge activation during language comprehension. *Journal of memory and language*, 82:118–132.

[4] Arevian, G. (2007). Recurrent neural networks for robust real-world text classification. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 326–329. IEEE Computer Society.

[5] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[6] Baroni, M. and Lenci, A. (2010). Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

[7] Baroni, M. and Zamparelli, R. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

[8] Bastianelli, E., Croce, D., Basili, R., and Nardi, D. (2013). Unitor-hmm-tk: Structured kernel-based learning for spatial role labeling. In *SemEval@ NAACL-HLT*, pages 573–579.

[9] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

[10] Bonial, C., Babko-Malaya, O., Choi, J. D., Hwang, J., and Palmer, M. (2010). Propbank annotation guidelines. *Center for Computational Language and Edu-cation Research Institute of Cognitive Science Uni-versity of Colorad at Boulder*.

[11] Brockmann, C. and Lapata, M. (2003). Evaluating and combining approaches to selectional preference acquisition. In *Proceedings of the tenth conference on European chapter*

of the Association for Computational Linguistics-Volume 1, pages 27–34. Association for Computational Linguistics.

[12] Carroll, J. D. and Chang, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319.

[13] Chersoni, E., Santus, E., Lenci, A., Blache, P., and Huang, C.-R. (2016). Representing verbs with rich contexts: an evaluation on verb similarity. *arXiv preprint arXiv:1607.02061*.

[14] Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

[15] Choi, E., Bahadori, M. T., Sun, J., Kulas, J., Schuetz, A., and Stewart, W. (2016). Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512.

[16] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[17] Clark, S. and Weir, D. (2001). Class-based probability estimation using a semantic hierarchy. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

[18] Coecke, B., Sadrzadeh, M., and Clark, S. (2010). Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*.

[Colah] Colah. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[20] Collobert, R. and Weston, J. (2011). Senna. *NEC Laboratories America, Inc*, 6.

[21] Dasigi, P. and Hovy, E. H. (2014). Modeling newswire events using neural networks for anomaly detection. In *COLING*, pages 1414–1422.

[22] Dowty, D. R. (1989). On the semantic content of the notion of 'thematic role'. In *Properties, types and meaning*, pages 69–129. Springer.

[23] Erk, K., Padó, S., and Padó, U. (2010). A flexible, corpus-driven model of regular and inverse selectional preferences. *Computational Linguistics*, 36(4):723–763.

[24] Ferretti, T. R., McRae, K., and Hatherell, A. (2001). Integrating verbs, situation schemas, and thematic role concepts. *Journal of Memory and Language*, 44(4):516–547.

[25] Geranmayeh, P. (2013). *Learning the Thematic Roles of Words in Sentences via Connectionist Networks that Satisfy Strong Systematicity*. PhD thesis, Applied Sciences: School of Computing Science.

[26] Gers, F. A. and Schmidhuber, E. (2001). Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340.

[27] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

[28] Greenberg, C., Demberg, V., and Sayeed, A. (2015). Verb polysemy and frequency effects in thematic fit modeling. In *Proceedings of the 6th Workshop on Cognitive Modeling and Computational Linguistics. Association for Computational Linguistics, Denver, Colorado*, pages 48–57.

[29] Hashimoto, K. and Tsuruoka, Y. (2016). Adaptive joint learning of compositional and non-compositional phrase embeddings. *arXiv preprint arXiv:1603.06067*.

[30] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

[31] Hitchcock, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *Studies in Applied Mathematics*, 6(1-4):164–189.

[32] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

[33] Hochreiter, S. and Schmidhuber, J. (1997a). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[34] Hochreiter, S. and Schmidhuber, J. (1997b). Ltsm can solve hard time lag problems. In *Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference*, pages 473–479.

[35] Klakow, D. and Peters, J. (2002). Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1):19–28.

[36] Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

[37] Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

[38] Landauer, T. K. and Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

[39] Le, Q. V., Jaitly, N., and Hinton, G. E. (2015). A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

[40] Lenci, A. (2011). Composing and updating verb argument expectations: A distributional semantic model. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 58–66. Association for Computational Linguistics.

[41] Liu, J., Wang, G., Hu, P., Duan, L.-Y., and Kot, A. C. (2017). Global context-aware attention lstm networks for 3d action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 7.

[42] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[43] Matsuki, K., Chow, T., Hare, M., Elman, J. L., Scheepers, C., and McRae, K. (2011). Event-based plausibility immediately influences on-line language comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 37(4):913.

[44] McCawley, J. D. (1976). *Grammar and meaning: papers on syntactic and semantic topics*. Academic Press.

[45] McRae, K., Spivey-Knowlton, M. J., and Tanenhaus, M. K. (1998). Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38(3):283–312.

[46] Melamud, O., Dagan, I., Goldberger, J., Szpektor, I., and Yuret, D. (2014). Probabilistic modeling of joint-context in distributional similarity. In *CoNLL*, pages 181–190.

[47] Memisevic, R. and Hinton, G. E. (2010). Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Computation*, 22(6):1473–1492.

[48] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[49] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

[50] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[51] Mikolov, T. and Zweig, G. (2012). Context dependent recurrent neural network language model. *SLT*, 12:234–239.

[52] Milajevs, D., Kartsaklis, D., Sadrzadeh, M., and Purver, M. (2014). Evaluating neural word representations in tensor-based compositional settings. *arXiv preprint arXiv:1408.6179*.

[53] Miller, G. A. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

[54] Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. In *ACL*, pages 236–244.

[55] Padó, U., Crocker, M. W., and Keller, F. (2009). A probabilistic model of semantic plausibility in sentence processing. *Cognitive Science*, 33(5):794–838.

[56] Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

[57] Raffel, C. and Ellis, D. P. (2015). Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*.

[58] Rieger, B. B. (1991). *On distributed representation in word semantics*. International Computer Science Institute Berkeley, CA.

[59] Ritter, A., Etzioni, O., et al. (2010). A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434. Association for Computational Linguistics.

[60] Rooth, M., Riezler, S., Prescher, D., Carroll, G., and Beil, F. (1999). Inducing a semantically annotated lexicon via em-based clustering. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

[61] Ruiz-Casado, M., Alfonseca, E., and Castells, P. (2005). Using context-window overlapping in synonym discovery and ontology extension. In *Proceedings of RANLP*, pages 1–7.

[62] Sayeed, A., Demberg, V., and Shkadzko, P. (2015). An exploration of semantic features in an unsupervised thematic fit evaluation framework. *Italian Journal of Computational Linguistics*, 1(1).

[63] Séaghdha, D. O. (2010). Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 435–444. Association for Computational Linguistics.

[64] Shen, D. and Lapata, M. (2007). Using semantic roles to improve question answering. In *Emnlp-conll*, pages 12–21.

[65] Sigtia, S., Benetos, E., Cherla, S., Weyde, T., Garcez, A., and Dixon, S. (2014). Rnn-based music language models for improving automatic music transcription.

[66] Socher, R. (2014). *Recursive deep learning for natural language processing and computer vision*. PhD thesis, Citeseer.

[67] Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., Potts, C., et al. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

[68] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

[69] Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

[70] Tang, D., Qin, B., and Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432.

[71] Tilk, O., Demberg, V., Sayeed, A., Klakow, D., and Thater, S. (2016). Event participant modelling with neural networks.

[72] Tsubaki, M., Duh, K., Shimbo, M., and Matsumoto, Y. (2013). Modeling and learning semantic co-compositionality through prototype projections and neural networks. In *EMNLP*, pages 130–140.

[73] Van de Cruys, T. (2014). A neural network approach to selectional preference acquisition. In *EMNLP*, pages 26–35.

[74] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

[75] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

[76] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A. J., and Hovy, E. H. (2016). Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489.

[77] Yin, W. and Schütze, H. (2014). An exploration of embeddings for generalized phrases. In *ACL (Student Research Workshop)*, pages 41–47.

[78] Yu, M. and Dredze, M. (2015). Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242.

[79] Zhou, J. and Xu, W. (2015). End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL (1)*, pages 1127–1137.

[80] Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., and Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

Appendix A

Evaluation data for *Patient, Verb* filler

(the sentences shown bellow are normalized with python *nltk* toolkit.)

- she avoided the annoying interest with a payment
- she avoided the annoying interest with a shortcut
- she avoided the annoying traffic with a shortcut
- she avoided the annoying traffic with a payment
- with a cauldron, she brew the medicinal potion for her twelve year old daughter
- with a kettle, she brew the medicinal potion for her twelve year old daughter
- with a kettle, she brew the medicinal tea for her twelve year old daughter
- with a cauldron, she brew the medicinal tea for her twelve year old daughter
- he caught the elusive baseball with a glove before it fell into someone else's hands
- he caught the elusive baseball with a net before it fell into someone else's hands
- he caught the elusive trout with a net before it fell into someone else's hands
- he caught the elusive trout with a glove before it fell into someone else's hands
- she colored her beautiful hair with a dye
- she colored her beautiful hair with crayons
- she colored her beautiful picture with crayons
- she colored her beautiful picture with a dye

Evaluation data for *Patient, Verb* filler

- with a joystick, he control the brand-new game that he bought yesterday
- with a remote, he control the brand-new game that he bought yesterday
- with a remote, he control the brand-new television that he bought yesterday
- with a joystick, he control the brand-new television that he bought yesterday
- with a white-out, she covered the minor error after she had discoverd the misspelling
- with a band-aid, she covered the minor error after she had discoverd the misspelling
- with a band-aid, she covered the minor scrape after she had discoverd the misspelling
- with a white-out, she covered the minor scrape after she had discoverd the misspelling
- with the scissors, she cut the expensive paper that she needed for her project
- with a saw, she cut the expensive paper that she needed for her project
- with a saw, she cut the expensive wood that she needed for her project
- with scissors, she cut the expensive wood that she needed for her project
- with a fork, she ate the homemade pasta that was stuffed with large piece of crab
- with a spoon, she ate the homemade pasta that was stuffed with large piece of crab
- with a spoon, she ate the homemade soup that was stuffed with large piece of crab
- with a fork, she ate the homemade soup that was stuffed with large piece of crab
- with a bottle, she fed the adorable infant who was born just two week ago
- with a bucket, she fed the adorable infant who was born just two week ago
- with a bucket, she fed the adorable pig who was born just two week ago
- with a bottle, she fed the adorable pig who was born just two week ago
- with a fireplace, she heated the frozen cabin that her grandma left to her
- with a oven, she heated the frozen cabin that her grandma left to her
- with a oven, she heated the frozen pie that her grandma left to her
- with a fireplace, she heated the frozen pie that her grandma left to her
- with a bat, he hit the dirty baseball really hard while playing in the backyard

- with a hammer, he hit the dirty baseball really hard while holding a beer in his other hand
- with a hammer, he hit the dirty spike really hard while holding a beer in his other hand
- with a bat, he hit the dirty spike really hard while playing in the backyard
- he held the antique photograph with a frame during his art class
- he held the antique photograph with a clamp during his art class
- he held the antique wood with a clamp during his art class
- he held the antique wood with a frame during his art class
- with a rifle, he killed the unfortunate deer that they had been purusing for an hour
- with a harpoon, he killed the unfortunate deer that they had been purusing for an hour
- with a harpoon, he killed the unfortunate whale that they had been purusing for an hour
- with a rifle, he killed the unfortunate whale that they had been purusing for an hour
- with a match, he light the cheap cigarette in the motel near the airport
- with a lantern, he light the cheap cigarette in the motel near the airport
- with a lantern, he light the cheap room in the motel near the airport
- with a match, he light the cheap room in the motel near the airport
- with the scissors, he opened the old package that he found in the basement
- with the canopener, he opened the old package that he found in the basement
- with the canopener, he opened the old soup that he found in the basement
- with the scissors, he opened the old soup that he found in the basement
- with an alarm, she protected the precious car that she purchased a month ago
- with an fence, she protected the precious car that she purchased a month ago
- with an fence, she protected the precious property that she purchased a month ago
- with an alarm, she protected the precious property that she purchased a month ago
- with a horse, he pulled the old-fashioned carriage from the barn to the park

Evaluation data for *Patient, Verb* filler

- with a pick-up truck, he pulled the old-fashioned carriage from the barn to the market
- with a pick-up truck, he pulled the old-fashioned trailer from the barn to the market
- with a horse, he pulled the old-fashioned trailer from the barn to the park
- with coins, she purchased the hand-made candy at the farmer's market
- with credit, she purchased the hand-made candy at the farmer's market
- with credit, she purchased the hand-made clothes at the farmer's market
- with coins, she purchased the hand-made clothes at the farmer's market
- with a rope, he secured the large boat properly so that no strong wind would blow it away from the dock
- with a lock, he secured the large boat properly so that no strong wind would blow it away from the dock
- with a lock, he secured the large door properly so that no one would break into his shed
- with a rope, he secured the large door properly so that it wouldn't fall over in his truck on his way home
- with a dish, he served the fabulous dessert following the main course last night
- with a mug, he served the fabulous dessert following the main course last night
- with a mug, he served the fabulous tea following the main course last night
- with a dish, he served the fabulous dessert following the main course last night
- with a shovel, he spread the fresh dirt all around the flower bed so that it made a nice mound
- with a knife, he spread the fresh dirt all around the flower bed so that it made a nice mound
- with a knife, he spread the fresh jam all around the his toast so that it covered the whole thing
- with a shovel, he spread the fresh jam all around world's largest loaf of bread
- with a hose, she washed her filthy car after she came back from the beach
- with the shampoo, she washed her filthy car after she came back from the beach

- with the shampoo, she washed her filthy hair after she came back from the beach
- with the hose, she washed her filthy hair after she came back from the beach
- with a rag, she wiped her greasy counter which became really dirty while she was making pancakes
- with a rakleenexg, she wiped her greasy counter which became really dirty while she was making pancakes
- with a rakleenexg, she wiped her greasy nose which became really dirty while she was cleaning the garage
- with a rag, she wiped her greasy nose which became really dirty while she was cleaning the garage
- with the paper, she wrapped the wonderful gift for her daughter who was coming to dinner that night
- with the tinfoil, she wrapped the wonderful gift for her daughter who was coming to dinner that night
- with the tinfoil, she wrapped the wonderful leftover for her daughter who was going back to her apartment
- with the paper, she wrapped the wonderful leftover for her daughter who was going back to her apartment