

## ABSTRACT

HSU, SHIOU TIAN. Rationale-based Neural Networks for Justifiable Relation Extraction. (Under the direction of Dr. Nagiza F. Samatova).

Relation extraction (RE) is the process of automatically extracting structured information from unstructured text, and is of critical interest for applications like knowledge base and question answering systems. As diversity and scope of RE-based systems have grown over the years, hand-crafted extraction rules became less feasible and autonomous data-driven RE systems are of more essentially to cope with the exploding growth of noisy user-generated text. However, while recent breakthroughs in neural network greatly benefited RE performance, the justifiability of the results remains a challenge to these data-driven approaches.

Rationales, as used in this thesis, are keywords and phrases that a human reader can reason upon to extract the desired relational information. That is to say, rationales can help illustrating the inference and justifying the results, and the essence of rationale-based RE is thus to locate and utilize these rationales to improve RE results and model interpretability. Another primary benefit that comes along is the identified rationales can become a correctable element that the users can manually adjust to refine the results. In this work, we focus on extracting relational information between target entities in sentences, as sentence is often the most fundamental unit in RE-based systems. We provide an example sentence in the following which expresses a Component-Whole relation between the entity-pair  $(e_1, e_2) = (\text{room}, \text{house})$ , and marked potential rationales a human would use in bold:

**Example sentence:** *“The disgusting scene was retaliation against her brother Philip who rents the [room] $e_1$  **inside** this apartment [house] $e_2$  on Lombard street.”*

As human readers often needs to grasp the general idea of the sentence first to determine the rationales, we emulate this process in this work. To begin with, we start with the task of understanding and representing the sentence as our first component of the thesis. To better cope with the noisy user-centric context most RE systems are now targeting, we position our sentence understanding component to focus on exploring grammar-independent methods for noisy on-line reviews. We propose a structure-independent ‘Gated Representation Alignment’ (GRA) model that blends a phrase-focused Convolutional Neural Network (CNN) approach with sequence-oriented Recurrent Neural Network (RNN). This alignment model allows the RNN to selectively include phrase information in a word-by-word sentence representation, and to do this without awareness of the grammatical structure. Empirical evaluation of GRA shows performance in parallel to state-of-the-art grammar dependent models, and up to 4.6% improvement (46.4% to 51.0%) comparing to other

grammar independent models.

We then propose the framework for rationale based RE in our second component. While rationales are seldom provided in datasets, our rationale-based RE model is designed to select rationales without prior rationale knowledge. Our RE framework consists of three stages: Generator, Selector, and Encoder. The Generator identifies candidate text fragments; the Selector decides which fragments can be used as rationales depending on the goal; and finally, the Encoder performs relation reasoning on the rationales. While the Encoder is trained in a supervised manner to classify relations, the Generator and Selector are designed as unsupervised models to identify rationales without prior knowledge, although they can be semi-supervised if given human annotated rationales. Experiments demonstrate that our approach outperforms state-of-the-art models on both extraction accuracy (88.0% to 89.5%) and interpretability (26.1% to 33.2%), and that our model is capable of extracting good rationales on its own as well as benefiting from labeled rationales if provided. Furthermore, we demonstrated that the model will change the prediction when new rationales are provided, which can be used in active-learning environment for better usability.

However, the rationale model can be very difficult to train due to its unsupervised nature at rationale discovery, and is susceptible to mode collapse where the Generator failed to select diversified rationales for different relations. In the final component, we work toward balancing the trade-off between extracting diversified rationales and achieving good rationale extraction results. To be more specific, we 1) discovered that the RNN used in the Generator as RNN is the major source of mode collapse, 2) replaced RNN with several simpler logistic regression components where each component learns from different features to diversify rationale selection, and 3) added a semi-supervised component cross reference rationales used in other sentences to stabilize rationale discovery. The updated rationale-based relation extraction framework further improved the extraction accuracy from 89.5% to 90.7%) and interpretability from 33.2 % to 53.2.

Rationale-based Neural Networks for Justifiable Relation Extraction

by  
Shiou Tian Hsu

A dissertation submitted to the Graduate Faculty of  
North Carolina State University  
in partial fulfillment of the  
requirements for the Degree of  
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2019

APPROVED BY:

---

Dr. Steffen Heber

---

Dr. Dennis R. Bahler

---

Dr. Kemafor Anyanwu Ogan

---

Dr. Nagiza F. Samatova  
Chair of Advisory Committee

## **DEDICATION**

To my wife Tzu Chen, who is also my best friend and mentor in life.

## **BIOGRAPHY**

Shiou Tian Hsu received his dual master degree of Management Information Systems and Technology Management from the dual degree program between National ChengChi University, Taiwan and University of Illinois at Urbana Chamapign(UIUC) on 2013. Shiou Tian started his interest in Natural Language Processing during his time at UIUC and decided to work on this area after two related projects. Shiou Tian is admitted to the Ph.D program in computer science at NCSU on 2014. During his Doctoral study, Shiou Tian had an internship at Facebook, Seattle working on News Feed and Search.

## **ACKNOWLEDGEMENTS**

The support and guidance from numerous people during this journey of Ph.D study were integral towards the work covered in this dissertation. I would like to thank my advisor, Dr. Nagiza Samatova for her help along the journey. I would also like to extend my gratitude to my committee members for their valuable advice: Dr. Stephen Heber, Dr. Dennis Bahler and Dr. Kemaphor Ogan. I am also very fortunate to have the opportunity to work with the group members of Dr. Samatova's research lab, especially Changsung Moon, Paul Jones, and Mandar Chaudhary. My fellow lab mates provided not only research expertise to overcome research challenges, but more importantly the support to go through the long journey of Ph.D study.

I have had the pleasure of collaborating with researchers at the Laboratory for Analytic Sciences (LAS), especially John Slankas on the KG project. This material is based upon work supported in whole or in part with funding from the Laboratory for Analytic Sciences (LAS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government.

## TABLE OF CONTENTS

<b>LIST OF TABLES</b> .....	<b>vii</b>
<b>LIST OF FIGURES</b> .....	<b>viii</b>
<b>Chapter 1 INTRODUCTION</b> .....	<b>1</b>
1.1 Rationale Based Relation Extraction .....	3
1.1.1 Modeling Unstructured Sentences .....	5
1.1.2 Rationale-based RE Framework .....	7
1.1.3 Multilevel-heuristics for Rationale Selection .....	7
1.2 Publications .....	8
<b>Chapter 2 Modeling Unstructured Sentences</b> .....	<b>9</b>
2.1 Introduction .....	9
2.2 Neural Networks .....	10
2.2.1 Feed-forward Neural Network .....	10
2.2.2 Convolutional Neural Network .....	15
2.2.3 Recurrent Neural Network .....	16
2.2.4 Recursive Neural Network .....	18
2.3 Methodology .....	21
2.3.1 Phrase Vector CNN .....	21
2.3.2 Soft-aligned RNN .....	23
2.3.3 Classification Layer and Regularization .....	24
2.4 Datasets and Experimental Setup .....	26
2.5 Results and Discussion .....	27
2.5.1 Effect of Soft-alignment .....	28
2.5.2 Structure-dependent Models .....	31
2.6 Conclusion .....	33
<b>Chapter 3 Rationale-Based Relation Extraction</b> .....	<b>34</b>
3.1 Introduction .....	34
3.2 Interpretable Networks and Generative Networks .....	37
3.2.1 Interpretable neural network .....	37
3.2.2 Generative Adversarial Network and Noise Contrastive Estimation .....	39
3.3 Adversarial Rationale Generation .....	41
3.4 Encoder, Generator and Selector .....	42
3.4.1 Encoder .....	42
3.4.2 Generator .....	43
3.4.3 Selector .....	44
3.4.4 Joint Objective and Adversarial Training .....	44
3.5 Experiments .....	46
3.5.1 Dataset .....	46
3.5.2 Experimental Setup .....	46
3.5.3 Results .....	47

3.5.4	Results with Ground Truth for Rationales . . . . .	50
3.6	Discussion . . . . .	52
3.6.1	Rationale Labeling . . . . .	52
3.6.2	End-to-End Training and Combining Components . . . . .	52
3.6.3	No-Rationale Cases . . . . .	52
3.6.4	Potential Uses . . . . .	53
3.7	Conclusion . . . . .	53
<b>Chapter 4</b>	<b>Stabilizing Rationale Selection . . . . .</b>	<b>54</b>
4.1	Introduction . . . . .	54
4.2	Related Work . . . . .	58
4.3	Proposed Model . . . . .	62
4.3.1	Encoder . . . . .	62
4.3.2	Generator . . . . .	63
4.3.3	Selector . . . . .	64
4.4	Experiments . . . . .	66
4.4.1	Dataset . . . . .	66
4.4.2	Experimental Setup . . . . .	66
4.4.3	Results . . . . .	67
4.5	Discussion and Future Goals . . . . .	68
4.5.1	Logistic Regressions and RNN . . . . .	68
4.5.2	Axillary Information . . . . .	69
4.5.3	Potential Applications: Noise-Reduction in Distant Supervision . . . . .	70
4.6	Conclusion . . . . .	71
<b>Chapter 5</b>	<b>Conclusion and Future Work . . . . .</b>	<b>72</b>
5.1	Conclusion . . . . .	72
5.2	Future Work . . . . .	74
<b>BIBLIOGRAPHY</b>	<b>. . . . .</b>	<b>76</b>

## LIST OF TABLES

Table 2.1	Accuracy of GRA and benchmarks. † denotes models that are trained on SST5 but sum the result of the softmax layer to obtain binary predictions; as stated in [Mou:2015], it is more difficult to obtain good results with this approach. . .	28
Table 2.2	Accuracy of GRA and other hybrids. . . . .	29
Table 2.3	Example sentences that are correctly predicted by GRA only. . . . .	31
Table 2.4	Example sentences that are incorrectly predicted by GRA only. . . . .	31
Table 2.5	Accuracy of GRA against structure dependent methods. † has same meaning as Table 2.1. . . . .	33
Table 3.1	10 Types of relations classes. Besides ‘Other’, all other relation types are bidirectional . . . . .	46
Table 3.2	Comparisons with benchmarking models . . . . .	49
Table 3.3	Comparison with proxy rationale models. . . . .	49
Table 3.4	List of words/phrases that are selected as rationales by our model. Additional words are denoted as ... if they are located between entities and rationales but are not selected as rationales. . . . .	50
Table 3.5	List of ground-truth rationales used . . . . .	51
Table 4.1	10 Types of relations classes. Besides ‘Other’, all other relation types are bidirectional . . . . .	66
Table 4.2	Comparisons with benchmarking models . . . . .	68
Table 4.3	Comparisons between variations and with rationale models. The numbers in the parentheses are confidence interval. Note that we trained a faster version of our model by reduced sampling for confidence intervals to make the model trained within a reasonable time. To be specific about the modified settings, we changed epoch from 25 to 15, and sampling times from 25 to 10. . . . .	68
Table 4.4	Detailed F1 score for each relations. . . . .	69
Table 4.5	List of words/phrases that are selected as rationales in test set by our model. Additional words are denoted as ... if they are located between entities and rationales but are not selected as rationales. . . . .	70

## LIST OF FIGURES

Figure 1.1	Parkinson disease related genes and diseases. Each center of a community is a disease that is related to genes that are related to the Parkinson disease. This graph is obtained from Wikidata, where entity-relations are annotated by on-line, anonymous co-authors. Entities are nodes in the image, and edges represent the relationship. Note that the edges here, although not visible, are all annotated by the relation between the entities. . . . .	2
Figure 1.2	Overall pipeline for rationale based relation extraction mapping to human process of relation extraction, along with how evaluations are performed. For a human reader, the RE process begins with understanding the general idea of the sentence, and then select rationales based on the sentence meaning and entities. We emulate this process by a sentence modeling module and a rationale-based RE module. The sentence modeling module, besides providing inputs to the rationale-based RE module, can be a stand-alone system for sentence classification tasks. . . . .	4
Figure 1.3	Comparison of rationales between [Lei16] and this work. Examples of [Lei16] is of the left, where each color indicates rationales of different aspect. Examples of rationale in this work in on the right, where each example begins with the relation type, words without entity tags are selected rationale from the model. Words in bold indicate overlapping rationale in different relation type. . . . .	5
Figure 1.4	Overview of the dissertation chapters and how they relate to each research questions. . . . .	6
Figure 2.1	Feed-Forward Neural Network with one-hot input layer. . . . .	12
Figure 2.2	Word vectors example. All words vectors are transformed from a 300-Dimensional space to 2-Dimensional space by TSNE. Words of similar semantic functions will be closer in the space. (TSNE is a dimension reduction tool that is commonly used for plotting word vectors . . . . .	13
Figure 2.3	Continuous Bag-of-Words(CBOW) and Skip-Gram. [Mik13] . . . . .	14
Figure 2.4	An example of Convolutional Neural Network generating n-gram features. CNN takes a sliding window approach that creates multiple small segments of the input. It allows multiple input points to be considered and interact at the same time, which is often useful for image classification where multiple pixels can be considered at the same time to identify image patterns. In natural language processing, CNN allows multiple words be considered at the same time, which can be considered as analyzing phrase information in the text. . . . .	17
Figure 2.5	An example of RNN. . . . .	17

Figure 2.6	An example of LSTM cell. The cell contains a forget, input and output gate. Different from the most simple RNN model, the LSTM cell does not directly take the output state from previous time step, instead it pass along a hidden state in each time step, outputs are finally obtained by passing hidden states to output gates. The benefit of splitting output and hidden state it can detach output from the cells directly. Finally, the forget gate will learn to forget portion of previous hidden state, and input gate will learn to include a portion of the latest input . . . . .	19
Figure 2.7	An example of GRU Cell. GRU cell combines the output and input gate to update gate. While it associates output directly to the cells, due to the reset gate, it can still avoid gradient vanishing efficiently. The major benefit of GRU over LSTM is GRU can be computed faster due to fewer gates. . . . .	19
Figure 2.8	An example of sentence parse tree with the example sentence “If you sometimes like to go to the movies to have fun, this movie is a good place to start”. We used Berkley parser in this example. . . . .	20
Figure 2.9	An example of Recursive Neural Network [Soc12b]. . . . .	21
Figure 2.10	GRA Framework and Details at Step i. . . . .	22
Figure 2.11	An example of CNN generating phrase vectors for word <b>blue</b> . . . . .	23
Figure 2.12	RNN cell in GRA. Each cell controls the attention scores for n-gram of different windows size. Each cells are connected, for example outputs from the cell for 2-gram will be used as inputs for the 3-gram cell. . . . .	25
Figure 2.13	Coverage of CNN and LSTM correct cases between GRA and GRA-without-alignment. . . . .	30
Figure 2.14	Figure on the left is the Venn diagram of sentences predicted correctly by GRA (our model) CNN, and RNN models. Figure on the middle is the same but using GRA without alignment. Finally, figure on the right is the Venn diagram considering only cases that are predicted correctly by GRA or GRA no-alignment. One can see that GRA outperforms the no-alignment version by covering more cases from CNN model and also utilize the combination of CNN and RNN to discover more sentences. Also, the two variations of GRA does not have much overlap. . . . .	30
Figure 2.15	Change of sentiment distribution when sentiment of sentence is manually reversed. Sentiment distribution is obtained by feeding the derived sentence vectors to the softmax layer. The sample sentence was a <b>positive</b> sentence: “If you sometimes like to go to the movies to have fun, this movie is a <b>good</b> place to start”. I replaced “a good” with “not a good” to reverse the sentiment of the sentence. . . . .	32

Figure 3.1	Model Structure with sample input sentence. Entities are given from the dataset. The Generator selects candidate rationales, and the Selector enumerates all possible combinations of candidates with entities and selects one ‘best performing candidate set’. An adversary Generator and Selector carry out the same process using a randomized approach. Candidate sets are then transformed into a vector representation by the Encoder and are evaluated against the ground-truth. Rewards are fed back to the Generator if the Encoder is able to identify candidate sets that are generated randomly, while the Selector is rewarded if the best performing rationale candidate set outperforms the one in the adversary. The best performing candidate set from Selector are finally considered as the extracted rationales, and then used in the Encoder to predict entity relation. In this example, the Generator first selected “scene”, “retaliation”, “inside” as candidates, then the Selector generates phrases using the entities - room and house, and the candidates. After the phrases are generated, it passes to the Encoder to be scored, and the best scored set is rewarded in the Selector, which in this case should be the phrase “room inside house”. Meanwhile, the Adversary Generator might randomly generate phrases by randomly selecting candidates, the Generator will be rewarded if it performs better than the Adversary Generator . . . . .	36
Figure 3.2	Attention model used in [bahdanau2014]. The model was originally used for language translation. Prior [bahdanau2014], language translation used a Encoder-Decoder approach that combines two Recurrent neural networks, where states for each word in the Encoder will be feed to the Decoder then attempts to generate corresponding translated word, a major drawback is the Encoder and Decoder is not aligned. That is, the Decoder will naively try to create a word that matches to every word in the original language. Attention model is introduced to weight outputs from Encoder prior pass to the Decoder, words with higher attention resembles the which words the Decoder should focus on. While it is used initially in language translation, it is then used in many different areas as it is efficient and simple. . . . .	38
Figure 3.3	Comparison of rationales between [Lei16] and this work. Examples of [Lei16] is of the left, where each color indicates rationales of different aspect. Examples of rationale in this work in on the right, where each example begins with the relation type, words without entity tags are selected rationale from the model. Words in bold indicate overlapping rationale in different relation type. . . . .	39
Figure 3.4	Comparison between GAN and NCE. The GAN model is on the left and NCE on the right. In GAN, there are two optimizable components: Generator and Discriminator. In NCE, there’s only a NCE model to optimize. GAN will optimize Generator and and Discriminator separately, and NCE will optimize the model using real data and noise data the same time. . . . .	40
Figure 3.5	F1 results using different intervention strategy. . . . .	51

Figure 4.1	<b>Left Figure:</b> We illustrate the relation of rationales of different perspectives in a simple way: rationales the Encoder favors $r_E$ , rationals generated by Generator $r_G$ and finally ground-truth rationales $r_T$ . There are multiple drawing forces that draws the Generator. <b>Right Figure:</b> The ideal results after training	55
Figure 4.2	<b>Left Figure:</b> The theoretical outcome of a successful result of our rationale-based relation extraction model - the Generator outputs diversified but convincing outputs to the Encoder. <b>Right Figure:</b> The outcome that happens frequently during our training process where the Generator decision factors collapsed to a single criteria(mode)	56
Figure 4.3	A Generator can converge at collapsed rationales but still lead to good classification results. <b>Left Figure</b> shows the rationale distribution of a collapsed and a well-converged Generator that uses a RNN. The well-converged Generator evenly selects rationales from a sentence, while the collapsed Generator selects rationales close to the beginning of the sentence. This is because the best scenario for Encoder is that some rationales are repetitively selected and optimized in every training iteration, which is difficult for the Generator. Eventually, the Generator learns to exploit position features since it is less diversified than words. <b>Right Figure</b> shows that a collapsed Generator does not necessarily deteriorate the classification performance.	56
Figure 4.4	<b>Left Figure:</b> The Issue of unbalanced Encoder and noise. The variety of rationales shrinks as the Generator collapsed to utilize only monotonous signals to decide rationales. <b>Middle Figure:</b> The issue of poor drawing force to good rationales, where the Generator converged to generate a poor collection of rationales. <b>Right Figure:</b> Results of insufficiently tuned model where both mode collapse and the poor drawing force happened.	56
Figure 4.5	Figure on the left shows the original Generator in previous chapter, and the right shows the improved multilevel heuristics Generator. The Word heuristic measures the relative closeness of each word in the sentence to every relation class. The Entity heuristic uses the target entities to regulate the Word heuristic. The Relation heuristic contains predictions of the relation from a standalone convolutional neural net model and is also used used to regulate the Word heuristic. The Word heuristic is then summed and weighted by the Syntax heuristic, which considers both, the word distance to entities, and Part-of-Speech labels. The Corpus heuristic is obtained by word-relation distribution in the corpus. The Rationale heuristic is the semi-supervised component which is based on rationales from the previous training iteration.	57
Figure 4.6	The major improvement lies in the Generator step by changing the Generator from a RNN to several simple feed-forward Neural Networks that optimizes against different level of text features. By specifically separating the optimization goal, we hypothesize it will alleviate mode collapse issue happened while using recurrent neural network where the Generator is solely guided by the Discriminator.	58

Figure 4.7	The basic generative adversarial network framework [Goo14]. The Generator will try to make fake samples based on some raw inputs, and the goal for Discriminator is to distinguish the fake samples from true samples. Take a text to image example where the text is "A yellow bird on a tree", the Generator will map the text to a generated image, then feed the generated image with the real image that has a yellow bird on a tree to the Discriminator. The Discriminator is trained using the true labels, and the Generator is rewarded/penalized based on successfully deceiving the Discriminator or not. . . . .	59
Figure 4.8	Example output from [Ree16]. The text is the input and the outputs are generated image that matches to the sentence meaning. . . . .	60
Figure 4.9	Mode collapse visualization from [Met16]. The data is a collection of 2D Gaussian Distributions, the final column is the ground-truth ( a dotted circle). The upper row shows the more expected results, where a well-trained GAN model will be able to produce results similar to the ground-truth. The bottom row represents a standard collapsed GAN where the Generator produces one type of data pattern (only points of a certain region in the ground-truth) in a period of training epochs. . . . .	60
Figure 4.10	VEEGAN model[Sri17]. $F_\theta$ represents the Reconstructor. Figure on the left shows when mode collapsed is happening, inverting the results of the Generator through Reconstructor $F_\theta$ should show a different distribution to the input data. Figure on the right shows Reconstructor can also help tune the Generator by serving as an autoencoder. . . . .	61
Figure 4.11	Wgan results[Arj17]. The figure shows in standard GAN the gradients at Discriminator side could vanish hence stop training. . . . .	62

## CHAPTER

# 1

# INTRODUCTION

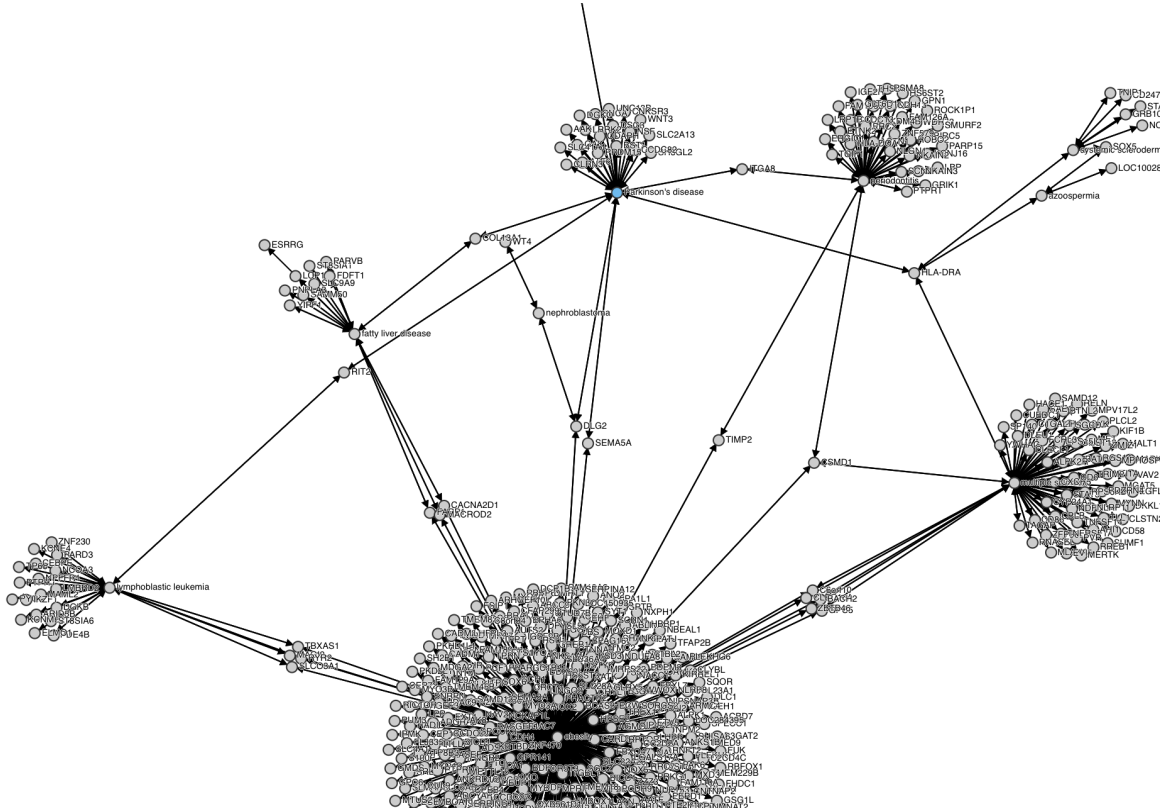
Relation extraction (RE) is the process of inferring the connection between target entities in text through natural language processing models to create relational information [Moe06; Sar08; Eic08; Ang15]. In other words, RE converts text information into connected pieces, allowing inference models or graph-based models to work on text data sources. For example, a RE system can parse news and build connections between entities of an event to provide an overview [Sar08; Mar13; Yam17]. Also, a diagnostics system can learn to connect syndromes with potential causes by scanning large medical corpus, which can help practitioners to avoid missing uncommon diseases<sup>12</sup> [Muz15]. We show an example of Wikidata, which is a large relational information database in Figure 1.1 as an example of human curated relational information. However, while still preferred in some cases for quality, manually curating relational knowledge from text became infeasible both in time and scope [BB11]. As of year 2017, there are more than 15M of text messages and 456K of tweet messages sent, 75K microblogs posted, and 600 wikipedia pages are being edited in every minute<sup>3</sup>. As a result, RE has drawn consistent interest from the research community.

Recently, neural network models has been widely adopted in RE research over the tradition rule based models due to their effectiveness [NG15b; Sah16; Zen15; MB16]. The major source of the effectiveness of neural network models come from their data-driven nature and the ability to

<sup>1</sup><https://www.microsoft.com/developerblog/2016/09/13/training-a-classifier-for-relation-extraction-from-medical-literature/>

<sup>2</sup><https://onlinelibrary.wiley.com/doi/full/10.1002/asi.23876>

<sup>3</sup><https://www.domo.com/learn/data-never-sleeps-5>



**Figure 1.1** Parkinson disease related genes and diseases. Each center of a community is a disease that is related to genes that are related to the Parkinson disease. This graph is obtained from Wikidata, where entity-relations are annotated by on-line, anonymous co-authors. Entities are nodes in the image, and edges represent the relationship. Note that the edges here, although not visible, are all annotated by the relation between the entities.

tailor enormous data by dense neuron layers. Besides improving effectiveness, several initiatives for improving interpretability and justifiability in neural networks in general has taken place [Dum09; Sim13; Ngu16; Mon17]. As interpretability of neural network improves, it allows human user to better disentangle and learn patterns from the complicated operations in neural network models, and also provides insights to correct erroneous results. While each model works differently, the general goal for interpretability is to identify the major contributors for each neuron or the final result. We demonstrate an example of the existing weight based neural network for RE [Wan16] in the following, where the values following each word shows the contribution of each word for inferring the relation. The example sentence expresses a Component-Whole relation between the entity pair  $\{room_{e_1}, house_{e_2}\}$ .

**Example sentence:** “The disgusting scene was retaliation against her brother Philip who rents the

*[room]e<sub>1</sub> inside this apartment [house]e<sub>2</sub> on Lombard street."*

**Weight-based models:** "*The<sub>0.02</sub> disgusting<sub>0.03</sub> scene<sub>0.1</sub> was<sub>0.02</sub> retaliation<sub>0.03</sub> against<sub>0.07</sub> her<sub>0.015</sub> brother<sub>0.02</sub> Philip<sub>0.04</sub> who<sub>0.08</sub> rents<sub>0.03</sub> the<sub>0.1</sub> [room]e<sub>1</sub> inside<sub>0.13</sub> this<sub>0.03</sub> apartment<sub>0.09</sub> [house]e<sub>2</sub> on<sub>0.01</sub> Lombard<sub>0.03</sub> street<sub>0.08</sub>."*

However, there still exist a gap to justifiability as human readers derives relational information differently. When trying to infer the relation between entities, a human reader examines the text and find several keywords that are self-expressing of the relation nature; other words are then simply treated as axillary information or even noise. These indicating words are referred as *rationales* in this work as per defined in [Zai07; Lei16; Zha16b]. We demonstrate an example of rationale(s) that would be used for a human reader in bold in the following sentence :

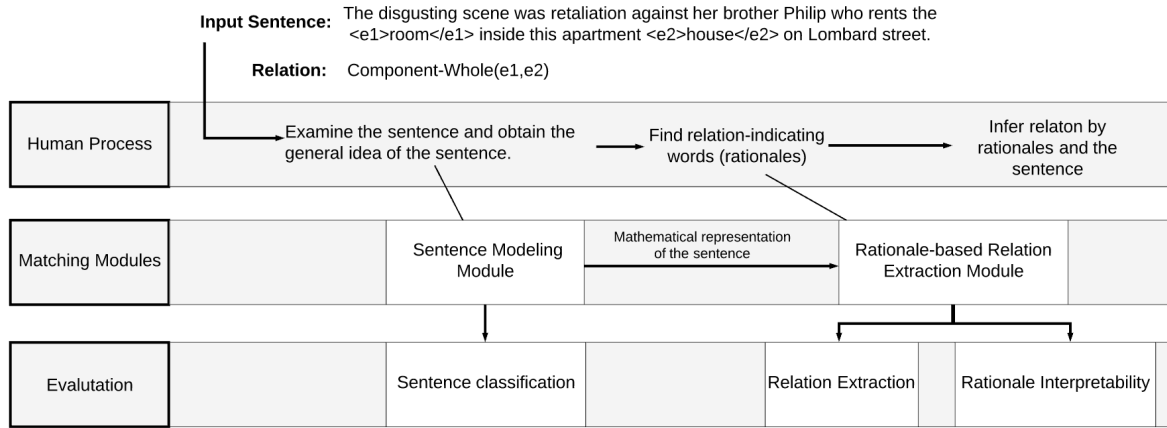
**Human intuition:** "*The disgusting scene was retaliation against her brother Philip who rents the [room]e<sub>1</sub> **inside this apartment** [house]e<sub>2</sub> ~~on Lombard street.~~"*

## 1.1 Rationale Based Relation Extraction

Rationale-based relation extraction, as introduced in this thesis, attempts to emulate how human derives relational information in neural networks. This provides a twofold benefit over existing methods. First, the output better relates to the interest of the user by showing the exact words used in the inference instead of weights, which means users can easily grasp the highlights in the text and gauge how justifiable the result is [Lei16]. Second, rationales offers a better interface for human users to interact with the model. For instance, to correct a false prediction made by the model manually, adjusting a few rationales is far more feasible and less trivial than tailoring weights for each word in the weight models. As a result, we posit that the rationale-based models for RE can help facilitate an exploratory and interactive approach towards extracting relational information from text.

In this thesis, we explore rationale-based models in the context of unstructured sentences. The focus on sentences is of the result that sentences are the finest source for relational information in text. In addition to structured sentences, we are more interested in the noisy and unstructured sentences as structured sentences are less common in the user-centric environment like social media or on-line reviews. We demonstrate the overall pipeline of the proposed rationale based approach in the following Figure 1.2.

Although rationale-based models are not first defined in this thesis, we are the first to explore rationale-based models for relation extraction. We notice two previous research that uses rationales for sentiment classifications for on-line reviews [Zha16b; Lei16]. Rationales, as per defined in those research, are essential pieces of text that directly relates to the classification goal, and can range from a word to a short sentence. In the first research[Zha16b], the researchers are interest in the



**Figure 1.2** Overall pipeline for rationale based relation extraction mapping to human process of relation extraction, along with how evaluations are performed. For a human reader, the RE process begins with understanding the general idea of the sentence, and then select rationales based on the sentence meaning and entities. We emulate this process by a sentence modeling module and a rationale-based RE module. The sentence modeling module, besides providing inputs to the rationale-based RE module, can be a stand-alone system for sentence classification tasks.

effect of using manually curated rationales along with the original sentence to improve sentiment analysis for on-line reviews. Rationales used in this research are a few short text segments like “The film, however, is all good” or “Now onto from hell’s appearances”. In the second research [Lei16], where the goal is aspect based sentiment analysis, rationales are considered as short sections of text that leads to different aspects in the review. The goal for rationales is to make sentiment analysis for each aspect in the review using a more related subset of words instead of the whole review, and also provide justification to the prediction. In effect, the goal of [Lei16] is to select a short segment of text that can replace the original text for different review aspects.

This work is different in two sense and posted different challenges. First, comparing to [Zha16b], we are intended to extract rationales without any prior knowledge, as large knowledge graph often consists of thousands of different relations so manually curating rationales is not applicable. Second, while in [Lei16] rationales are well diversified between different aspects of the reviews, rationales in RE are often shared between different relations, like the word “of” is commonly used for Member-Organization, Message-Topic and Component-Whole relations (Figure 3.3). The lack of diversified rationales leads to potentially over exploiting certain rationales for prediction. Over exploitation of a small collection of rationales can dampen the justifiability as the model output the same rationale regardless the relation, and thus requires intensive tuning of the model. This issue is often noted as *mode-collapse* in related research.

We address the need for rationale-based RE on unstructured text by exploring a few research objectives in the context of sentence modeling and relation extraction:

<p>EXAMPLE 1. a beer that is not sold in my neck of the woods , but managed to get while on a roadtrip . poured into an imperial pint glass with a <b>generous head that sustained life throughout</b> . nothing out of the ordinary here , but a good brew still . body <b>was kind of heavy</b> , but <b>not thick</b> . the hop smell was excellent and enicing . very drinkable</p> <p>EXAMPLE 2: a : poured a <b>nice dark brown with a tan colored head about half an inch thick</b> , nice <b>red/garnet accents when held to the light</b> . <b>little clumps of lacing all around the glass</b> , not too shabby . not terribly impressive though s : smells <b>like a more guinness-y guinness really</b> , there are some roasted malts there , signature guinness smells , less burnt though , a little bit of chocolate ... m : relatively thick , it is n't an export stout or imperial stout , but still is pretty heavy in the mouth , <b>very smooth</b> , <b>not much carbonation</b> . <b>not too shabby</b> d : not quite as drinkable as the draught , but still not too bad . i could easily see drinking a few of these .</p>	<p><b>OF</b>  Component-Whole: &lt;e1&gt;configuration&lt;/e1&gt; <b>of</b> ... &lt;e2&gt;elements&lt;/e2&gt;  Member-Collection: &lt;e1&gt;crowd&lt;/e1&gt; <b>of</b> &lt;e2&gt;candidates&lt;/e2&gt;.  Message-Topic: &lt;e1&gt; speech was summary <b>of</b> &lt;e2&gt; ... problems</p> <p><b>In</b>  Entity-Destination:&lt;e1&gt;guns&lt;/e1&gt; ... <b>in</b> ...&lt;e2&gt;safe&lt;/e2&gt;  Member-Collection:&lt;e1&gt;sergeant&lt;/e1&gt; <b>in</b> &lt;e2&gt;army&lt;/e2&gt;</p> <p><b>BY</b>  Cause-Effect: &lt;e1&gt;dips&lt;/e1&gt; caused <b>by</b> ... &lt;e2&gt;y-rays &lt;/e2&gt;  Product-Producer:&lt;e1&gt;firm&lt;/e1&gt; co-founded <b>by</b> ... &lt;e2&gt;head &lt;/e2&gt;</p>
--	--

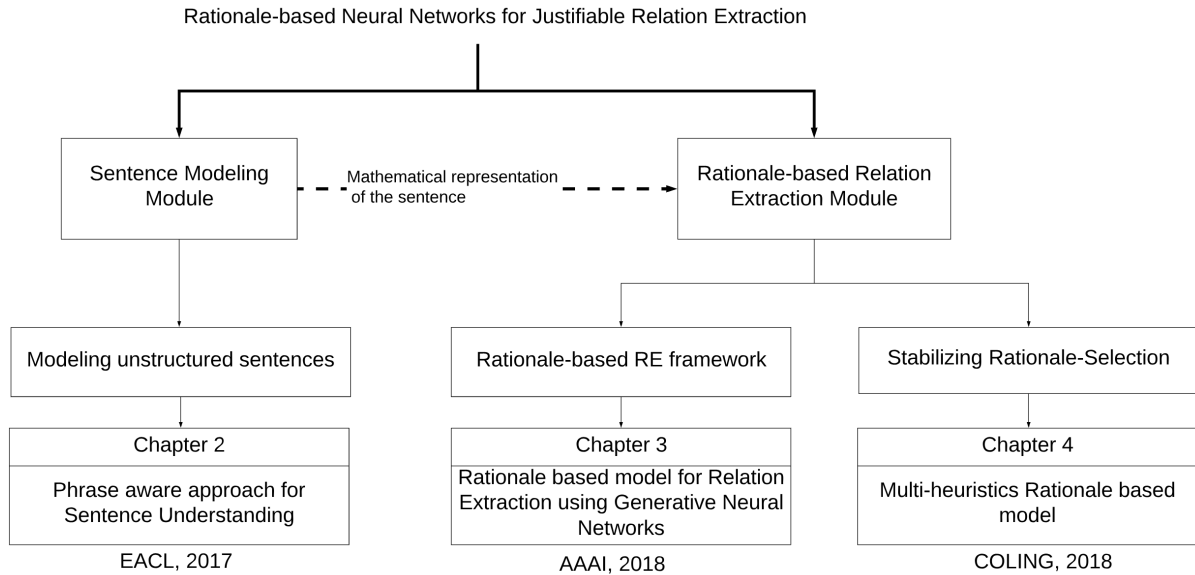
**Figure 1.3** Comparison of rationales between [Lei16] and this work. Examples of [Lei16] is of the left, where each color indicates rationales of different aspect. Examples of rationale in this work in on the right, where each example begins with the relation type, words without entity tags are selected rationale from the model. Words in bold indicate overlapping rationale in different relation type.

1. Developing sentence modeling algorithms that is structure-independent but performs on par with structure-dependent models.
2. Develop the framework for rationale-based RE that shows rationale along with the desired relational information, and allows users to examine and interact with the rationales.
3. Develop stabling and de-noising methods to improve rationale-selection to mitigate mode-collapse challenge.

Figure1.4 illustrates how the chapters are connected and related to the above research objectives. Our second chapter is the sentence analyzing module that understands and represent sentence in mathematical format for further process. We propose the method in our second chapter to better cope with unstructured text from crowd generated text sources. Our third chapter is the overall structure of the rationale based models. Finally, the forth chapter, is intended to improve model stability and diversify rationale selections in this rationale-based model.

### 1.1.1 Modeling Unstructured Sentences

State-of-the-art sentence modeling methods typically leverage distributed embedding models that are able to represent words as real valued vectors to capture their semantic characteristics. The word embeddings can then be combined into sentence embeddings to represent the sentence semantics. There are in general three strategies for combining the word vectors: n-gram models, sequential models and tree models. Of these, the best results have been obtained using tree models, which use sentence syntactic trees originating from grammar parser to help construct sentence



**Figure 1.4** Overview of the dissertation chapters and how they relate to each research questions.

embeddings. However, noisy text (such as found in on-line reviews) does not always contain much grammatical structure, which reduces the effectiveness of tree models. Hence it is important to study structure-independent models.

Much recent research into structure-independent n-gram CNN models attempts to build comprehensive sentence embeddings by identifying the most influential n-grams of different semantic aspects. However, while these methods are effective at exploring the regional syntax of words, they are unable to account for order-sensitive situations, where the order of words is critical to the meaning.

On the other hand, sequential models based on RNN build sentence embeddings using a *global cell* that reads one word at a time. The cell contains an update function that uses the most recent word to update sentence embeddings, while maintaining some memory of previously seen words. Recent extensions of RNN cells, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), better enable the cell to memorize and forget information that is pertinent to the meaning of the sentence. However, it is not clear how much phrase-level information is captured since the RNN cells are optimized from a whole-sentence perspective.

In this chapter, we propose a hybrid CNN-RNN framework to model relationships between phrases and word sequences in each sentence. In the framework, we added a soft-aligning layer that provides an adaptive mechanism for RNN to ‘peek’ into relevant n-grams generated by a CNN and selectively include them. We call our model *Gated Representation Alignment* (GRA) since we implement soft-alignment using a group of Gated Recurrent Units. Similar to CNN and RNN

approaches, GRA requires no explicit structural information about the sentence, making it adaptable to noisy text.

### **1.1.2 Rationale-based RE Framework**

In this chapter, we propose the overall framework for rationale-based RE that aims to improve interpretability of neural models by imitating the logic used by human annotators. The general idea is to select rationales, and then utilize these rationales for inferring the relation between the target entities.

Comparing to state-of-the-art supervised RE models, the major challenge of this work is to find rationales without any prior human knowledge. In other words, this work is a hybrid of unsupervised - rationale selection, and supervised - relation extraction tasks. As a result, we adopted the Generative Adversarial Network (GAN) that is well-know in it's ability to generate matching images to descriptive sentences. GAN models are typically a conjunction of a generative component and a discriminative component. We adopted this mindset, and came up with a three-stage Generator-Selector-Encoder approach. The Generator first enumerates candidate text fragments from the sentence using a Recurrent Neural Network (RNN) model, then the Selector extracts fragments that are informative for determining the relation and passes these on to the Encoder to make final relation predictions.

Besides answering the question if rationales are useful in both prediction accuracy and interpretability, we are also interested in answering how will the model change if rationale information is provided. In the cases where manually curated rationales are given, the Generator and Selector can become supervised model and learn to extract rationals based on prior knowledge.

### **1.1.3 Multilevel-heuristics for Rationale Selection**

Finally, we attempt to address a mode collapse issue in our model that causes instability by a multi-heuristics approach. Mode-collapse issue originated from the KL-divergence (Kullback-Leibler) that are most commonly used in GAN, where the model suffers from maximizing the output (prediction accuracy) and diversifying the generated materials (rationale diversity). In essence, a model that suffers from mode collapse will either constantly selecting the same rationales regardless the context, or achieves only mediocre prediction performance by forcefully using diversified rationales . In this chapter, we attempt to generalize rationale selection process by including multiple views of the sentence. Each different view is considered as a heuristics of different level, ranging from word level, sentence level and finally corpus and data source level.

## 1.2 Publications

This section lists publications and submissions associated with or enabled by the work described in this dissertation.

1. Title: A Hybrid CNN-RNN Alignment Model for Phrase-Aware Sentence Classification  
Publication: European Chapter of Association of Computational Linguistics (EACL), 2017, Valencia, Spain.  
Authors: Shiou Tian Hsu, Changsung Moon, Paul Jones, Nagiza F. Samatova
2. Title: An Interpretable Generative Adversarial Approach to Classification of Latent Entity Relations in Unstructured Sentences  
Publication: AAAI Conference on Artificial Intelligence, 2018, New Orleans, Louisiana.  
Authors: Shiou Tian Hsu, Changsung Moon, Paul Jones, Nagiza F. Samatova
3. Title: Multilevel Heuristics for Rationale-Based Entity Relation Classification in Sentences  
Publication: International Conference on Computational Linguistics (COLING), 2018, Santa Fe, New Mexico.  
Authors: Shiou Tian Hsu, Mandar Chaudhary, Nagiza F. Samatova

# MODELING UNSTRUCTURED SENTENCES

## 2.1 Introduction

Sentence modeling is the task of modeling and representing the sentences in comparable mathematical forms[Kal14a; LM14]. It is an important enabler for many applications requiring a degree of semantic comprehension, for example sentiment analysis or document classification. In this thesis, it also provides guidance to select rationales. Advancement in sentence modeling area largely relates to the ability to represent and combine words in sentence. As more efficient algorithms for distributed embedding models has been proposed,[Mik13], which discover semantic relations between words and represent words as real-valued vectors, most recent research in the area has shift to neural network based models [Kim14a; Kal14b; Soc12a; Tai15; LM14]. These state-of-the-art sentence modeling methods typically attempt to combine the word vectors to form a sentence representation following three strategies: n-gram models, sequential models and tree models. Of these, the best results have been obtained using tree models [Mou15; Tai15], which use sentence syntactic trees originating from grammar to help construct sentence embeddings. However, noisy text (such as found in on-line reviews) does not always contain much grammatical structure, which reduces the effectiveness of tree models. Hence it is important to study also structure-independent models.

Much structure-independent sentence modeling research has been into n-gram based CNN models, which are originally used in image classification [Kal14a; Yu14a; YS15; Kim14a; Zha16a]. As CNN learns how to group pixels to objects for image classification, it works as a sliding window on sentences as can be seen as learning to group words into expressive n-grams. Sentence representation are then finally compiled by combining the most influential n-grams of different semantic aspects together. However, while these methods are effective at exploring the regional syntax of words, they are unable to account for order-sensitive situations, where the order of words is critical to the meaning.

On the other hand, sequential models based on RNN [Gra13; Sut14; Pal16] build sentence embeddings using a *global cell* that reads one word at a time. The cell contains an update function that uses the most recent word to update sentence representation, while maintaining some memory of previously seen words. Recent extensions of RNN cells, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) [Cho14], better enable the cell to memorize and forget information that is pertinent to the meaning of the sentence. However, it is not clear how much phrase-level information is captured since the RNN cells are optimized from a whole-sentence perspective.

In this chapter, we introduce our work of a hybrid CNN-RNN framework to model relationships between phrases and word order in the sentence. In this framework, we added a soft-aligning layer that provides an adaptive mechanism for RNN to ‘peek’ into relevant n-grams generated by a CNN and selectively include them. The model is called *Gated Representation Alignment* (GRA) since soft-alignment was implemented using a group of Gated Recurrent Units. Similar to CNN and RNN approaches, GRA requires no explicit structural information about the sentence, making it adaptable to noisy text.

Due to difficulty in directly assessing the quality of the mathematical forms of the sentences, following other sentence modeling research, we evaluate this work through classification experiments. In the experiments, GRA outperforms the common RNN baseline by 4.6% when classifying fine-grained sentence level sentiment-classification datasets. Furthermore, GRA achieves comparable results to structure-dependent models on well-structured sentences, but outperforms those models when the sentences are less-structured. Finally, further analysis against baselines shows the alignment mechanism in GRA is the key to combine the power of CNN and RNN approaches, as CNN-RNN hybrid does not always guarantee good results.

## 2.2 Neural Networks

### 2.2.1 Feed-forward Neural Network

Most recent breakthroughs for sentence modelings are derived from the introduction of word vectors and related neural network approaches. Most neural network models are extensions of the

feed-forward neural network, including the three other types of neural network introducing in the following subsection. Hence a feed-forward neural network can be considered as the basic form of neural networks.

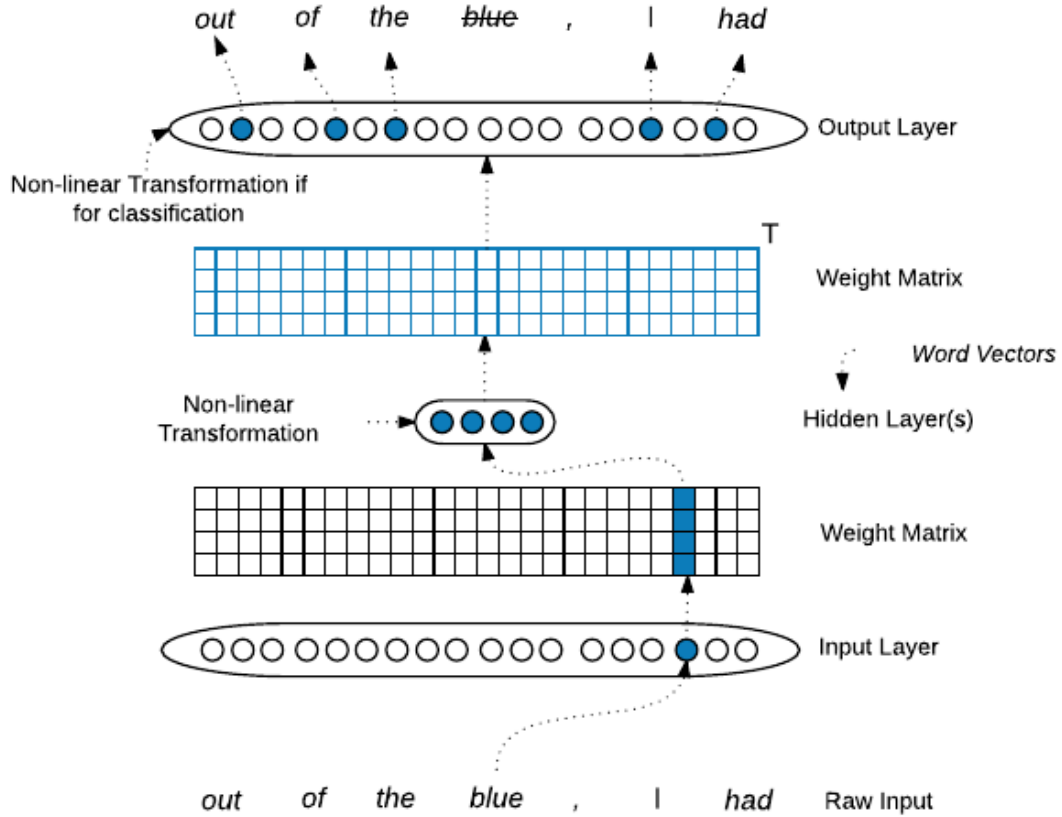
In the shallowest form of a feed-forward neural network as shown in Figure 2.1 contains three layers: 1) input, 2) hidden and 3) output layer. For most natural language tasks, the input layer is a one-hot encoding table that encodes a token (i.e. a character or word) to an index. Between each layer, there is a transition weight matrix to forward the vector(embeddings) from the previous layer to the next. For all hidden layers, it performs a non-linear transformation using an activation function on the received embeddings. An example of the activation function is the sigmoid function or the hyperbolic tanh function. The non-linear transformation is one of the most important functions within neural networks as it enables the hidden layer to further differentiate the embeddings by stacking layers. A multi-layered neural network without activation function can be directly approximated by a single layer neural network. The output layer, depending on the task, outputs the embedding as-is for regression tasks or perform non-linear transformation for classification tasks. A node is called a neuron in neural networks, where it encapsulates the activation function and transition matrix for the given node pairs.

One of the major advancement in natural language processing derives from neural network is word embeddings [Mik13] (Figure 2.2), which they leverage an unsupervised feed-forward network to embed a word token to a vector space. There are two different but very similar approaches for word vectors: Skip-Gram and Continuous Bag-of-Words (CBOW). Both models work under the assumption that semantically related words will co-occur closely in text more often than words of distant semantics. By using a sliding window approach, a Skip-Gram model will iteratively use the middle word in the window to predict other words in the window (similar to the example in right of Figure 2.3). Continuous Bag-of-Words is very similar but reversed the direction by using all neighboring words to predict the middle word instead.

### **2.2.1.1 Optimizing Neural Networks**

The most common way to train a neural network is through gradient descent and back propagation. During training, the output layer receives gradients based on a loss function measuring the error between the ground-truths and model's prediction. The gradients can then be propagated to the previous layer using the chain rule to update the weight matrices.

We begin with the most basic gradient descent algorithm and back propagation, then explain how different extensions overcome the enlisted challenges. In the most basic gradient descent model, model outputs are compared with the provided ground-truths. For regression models, the gradients are often the Euclidean distance between the output and given value, for example prediction of daily stock prices and the actual closing price. In classification models, outputs are often converted



**Figure 2.1** Feed-Forward Neural Network with one-hot input layer.

to binary or multi-classes outputs by logistic functions or softmax functions (Equation 2.1), and gradients are obtained by cross-entropy against the given labels. After gradients are obtained, the model will propagate the gradient back into the model layer-by-layer (Equation 2.2).  $J$  represents the gradient,  $\theta$  as parameters used in the network,  $o$  the output of the network,  $net$  as the input of the network, and finally  $\eta$  as a penalizing learning weight. Each layer will consume a portion of gradient as to update the parameters in the layer.

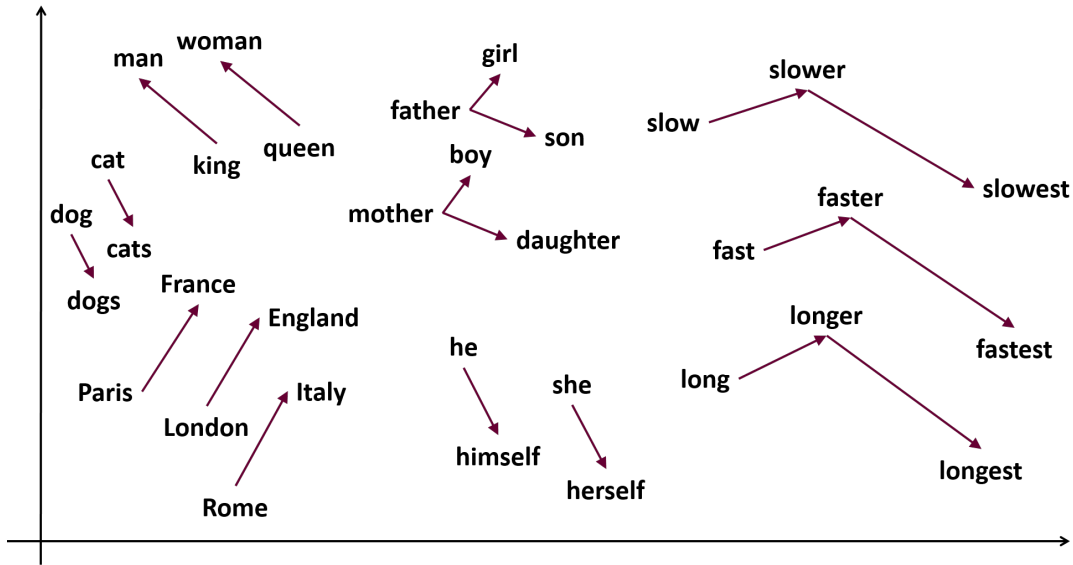
$$F(x_j) = \frac{e^{x_j}}{\sum_{k=1}^K e^{x_k}} \quad (2.1)$$

$$\nabla_{\theta} J(\theta) = \frac{\partial J}{\partial \theta} = \frac{\partial E}{\partial o} \frac{\partial o}{\partial net} \frac{\partial net}{\partial \theta} \quad (2.2)$$

$$new \theta = \theta - \eta \nabla_{\theta} J(\theta)$$

While gradient descent is the most common method used in neural networks, there are several extended approach of gradient descent to overcome the following two challenges:

1. Difficulty in searching global optimum: Gradient descent in neural network is often considered

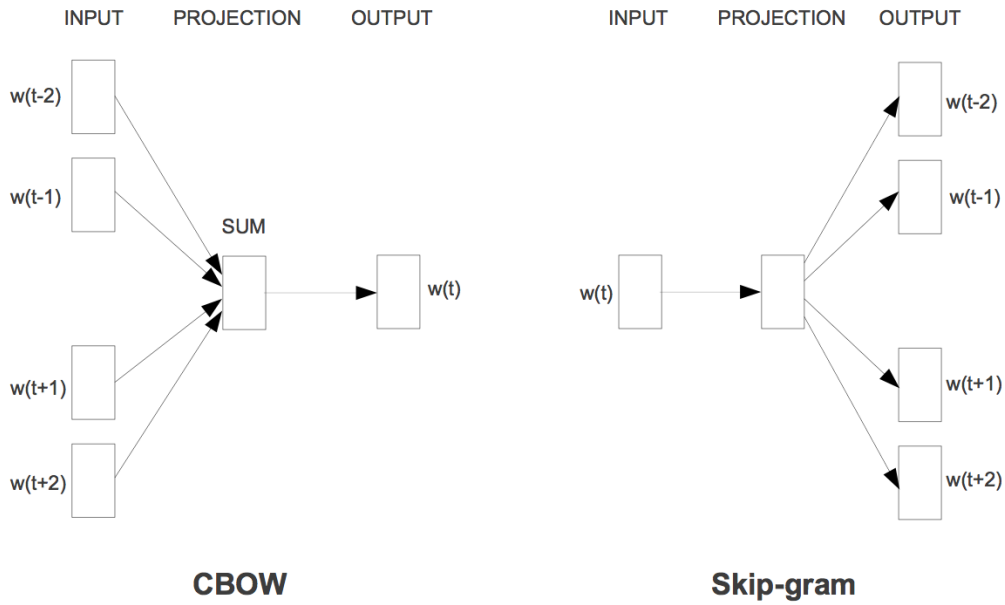


**Figure 2.2** Word vectors example. All words vectors are transformed from a 300-Dimensional space to 2-Dimensional space by TSNE. Words of similar semantic functions will be closer in the space. (TSNE is a dimension reduction tool that is commonly used for plotting word vectors)

as a Maximum Likelihood Method(MLE), where the goal is to find a set of variable settings that optimized the loss function. However, when training a neural network model, there often exist several pit holes - local optimum where the loss is minimized in a certain range of variable settings. For large pit holes, it creates challenges for gradient descent because there are no reasonable directions for variables in the pit holes to optimize the loss functions.

2. Hyper-parameter settings: In most gradient descent models, there exist several hyper parameters that are required from the researcher. For example, learning rate is the most common manually tuned parameter which controls the degree of gradient feedbacks on variables. Setting these hyper-parameters often requires expertise and experiments, and can bias comparison between research due to the extend of fine-tuning required for these hyper-parameters.

One of the ideas to deal with the first challenge in gradient descent is to take the momentums of previous updates into account [Qia99] as shown in Equation 2.3. In momentum approach, the model will record the direction of previous updates  $v_{t-1}$ , and penalize  $v_{t-1}$  by a  $\gamma$  factor. Momentum approach provides a good boost in speed to overcome oscillation in gradient decent that is often seen around local optimums. Extended approach like Nesterov update [Nes; Sut13] utilize momentum but added a correction force to avoid over responsiveness of gradient signals. While it does not directly tackles the second challenge, extending research has published regarding how to set or learn the hyper-parameters [Mac15].



**Figure 2.3** Continuous Bag-of-Words(CBOW) and Skip-Gram. [Mik13]

$$\begin{aligned}
 v_t &= \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta) \\
 \text{new } \theta &= \theta - v_t
 \end{aligned}
 \tag{2.3}$$

Another gradient descent optimization method called Adagrad [Duc11] approaches to the learning rate issue (Equation 2.4). In Adagrad, learning rate becomes adaptive to the gradient signals. For parameters that are updated frequently, Adagrad models will assign lower learning rate and vice versa for uncommon updates. A more intuitive idea is that the model considers frequent updated parameters as less sensitive signals and thus penalized it by assigning smaller learning rate.

$$\begin{aligned}
 g_{t,i} &= \nabla_{\theta} J(\theta_{t,i}) \\
 \text{new } \theta_{t,i} &= \theta_{t,i} - \frac{\eta}{\sqrt{\sum g_i^2}}
 \end{aligned}
 \tag{2.4}$$

While Adagrad preserves the average of previous gradients updates to modify learning rate, a common issue in Adagrad is that as training proceeds, learning rate is often over discriminated for frequent updated parameters, resulting those parameters stagnant in training regardless the magnitude of the gradient signal. This leads to the idea of AdaDelta [Zei12], where the adaptive

learning rate is penalized by a decaying averaged factor of previous updates (Equation 2.5). Besides overcoming learning rate diminishes, AdaDelta also removed manually set learning rate from the formula and relies purely on past gradients, which is helpful to avoid biased learning rate.

$$\begin{aligned}
 E[g^2]_t &= \gamma E[g^2]_{t-1} + (1-\gamma)g^2 \\
 \text{new } \theta_t &= \theta_t - \Delta\theta_{t,i} \\
 \Delta\theta_{t,i} &= -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t
 \end{aligned} \tag{2.5}$$

Finally, Adam [KB14] is a combination of Adagrad and Momentum (Equation 2.6). In Adam, learning rate and gradient signals are amplified by previous update directions and penalized by the sum of previous gradients. AMSGrad[Red18] is later proposed to use the max of previous gradient signals as benchmark instead of average to better respond to more extreme gradient signals.

$$\begin{aligned}
 m_t &= \beta_1 m_{t-1} + (1-\beta_1)g_t \\
 v_t &= \beta_2 v_{t-1} + (1-\beta_2)g_t^2 \\
 \hat{m}_t &= \frac{m_t}{\beta_1^t} \\
 \hat{v}_t &= \frac{v_t}{\beta_2^t} \\
 \text{new } \theta_t &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t
 \end{aligned} \tag{2.6}$$

While neurons (nodes) in adjacent layers in a feed-forward neural network are fully connected, there are several extensions of neural network where layers are not always fully connected. The way how nodes are connected are the major difference between a feed-forward neural network with the following three types of neural network.

### 2.2.2 Convolutional Neural Network

Convolutional neural network (CNN) [LeC98] is an extension of neural network originally intended for image identification [Kri12]. In CNN, images are fed into input layer using raw pixels and color value like RGB color scale. Each neuron in the hidden layers, instead of fully connecting to all neurons in the previous layer, connects only to a small region of neurons. Within each layer, the neurons share a group of filters and apply them to all inputting embeddings. A different way to see this is each filter are designated to recognize certain patterns, but regardless of the exact coordinates of the data. For example, if a filter is good at recognizing a leftward curve, CNN will break the image down into smaller chunks, and apply this filter throughout the image to identify all leftward curves.

This step is often called convolutional layer in CNN research.

Similar to feed-forward neural networks, CNN also applies a non-linear transformation on each layer. However, prior the transformation, CNN often applies pooling to downsize the output data. There are several different methods for pooling, like max-pooling which reports only the max values within each region, or average-pooling which reports the mean of region. Pooling stands on the assumption that recognizing a pattern does not always requires the finest detail of the image but instead an approximation, which helps greatly to decrease the size of representation at each layer.

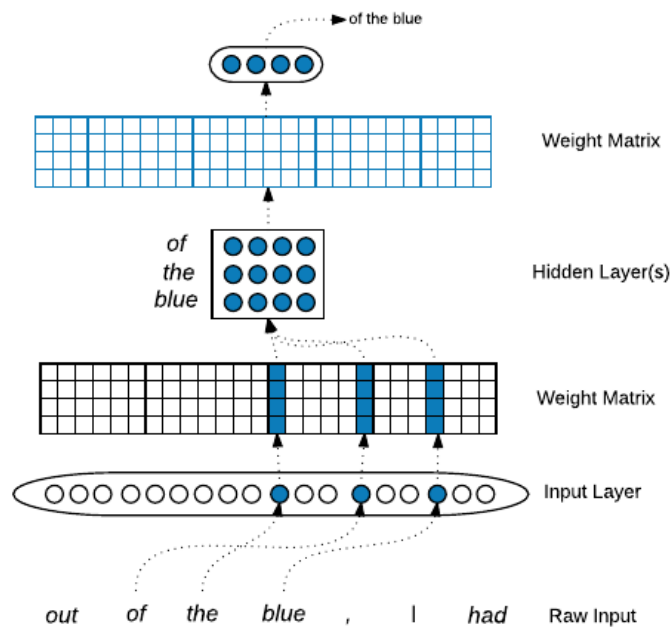
While input and hidden layers are often partially connected, the output layers are mostly fully connected to the previous layer. However, based on the nature of the targeted data, hidden layers can sometimes be fully connected to neighboring layer. For image using RGB or similar color coding system that has more than one color, a common approach is to treat each color as a feature and separate the colors at the input layer and combine them at a later stage.

While CNN excels at image identification, [Kim14a] successfully applied it to sentence modeling (Figure 2.4). To create a 2-dimensional data input for CNN, [Kim14a] utilizes word vectors of each word, and use the word sequence as the height-axis, and word vectors as width-axis. In the convolution process, each filter is designed to extract features from n-gram of words, hence the filter is of height n, where n belongs to [3,4,5] in [Kim14a], and width as the size of word vectors. Another breakthrough from [Kim14a] is they treat word vectors learned from different sources like different colors in RGB color coding, CNN trains a separate model for each word vector source and finally combines at the output layer. Further extension of this approach like [YS15] and [Zha16a] provided different ways to combine word vector sources.

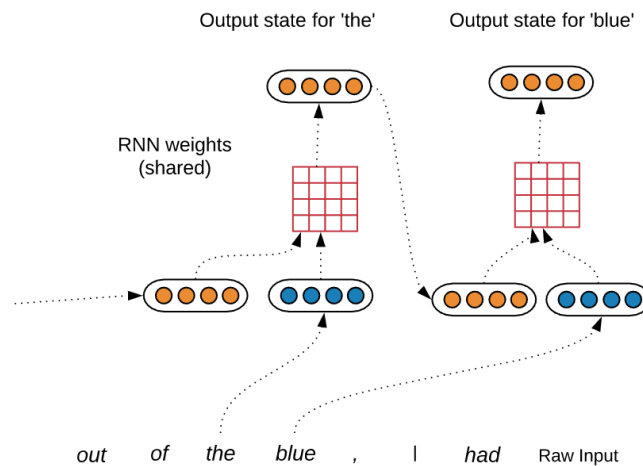
However, the performance of the convolution layer is susceptible to the distance between input neurons. While pixels are highly related with nearby pixels, words in sentences are often found not closely related with nearby words but instead distance words. In [Kal14a] approach, they came up with a generalized framework for the pooling stage allowing distance features to be interactive. Another argument for CNN models for sentence modeling is that word order is not specifically considered, thus the model will consider 'cat climbs tree' and 'tree climbs cat' as similar sentence.

### **2.2.3 Recurrent Neural Network**

Recurrent Neural Network(RNN) (Figure 2.5) is an extension of neural networks for sequential data. Using the CNN sentence modeling example, where a sentence is laid out in 2-dimensional form for the input layer, in RNN the sentence is break down into words, and the input layer takes in one word vector and a state vector at a time. A state vector is the result of the output layer in RNN at a given time, and is iteratively feed into the RNN model again with the next word. By recording information of the sequence using the state vector and use it as a moderator, RNN is able to interpret new information based on all previously seen sequence.



**Figure 2.4** An example of Convolutional Neural Network generating n-gram features. CNN takes a sliding window approach that creates multiple small segments of the input. It allows multiple input points to be considered and interact at the same time, which is often useful for image classification where multiple pixels can be considered at the same time to identify image patterns. In natural language processing, CNN allows multiple words be considered at the same time, which can be considered as analyzing phrase information in the text.



**Figure 2.5** An example of RNN.

Training for RNN is very much similar to feed-forward neural network, but rolled out the back propagation process into N stages, where N equals to the length of the input data. However, gradients could become faded if the propagated distance become overly-long, and the model stops updating. This problem is called gradient vanishing and is commonly seen in RNN models. Improved version of RNN intended to solve this by replacing the layers by cells. A cell generally contains an input, hidden and output gate that are designed to scale gradients accordingly to avoid gradients from vanishing.

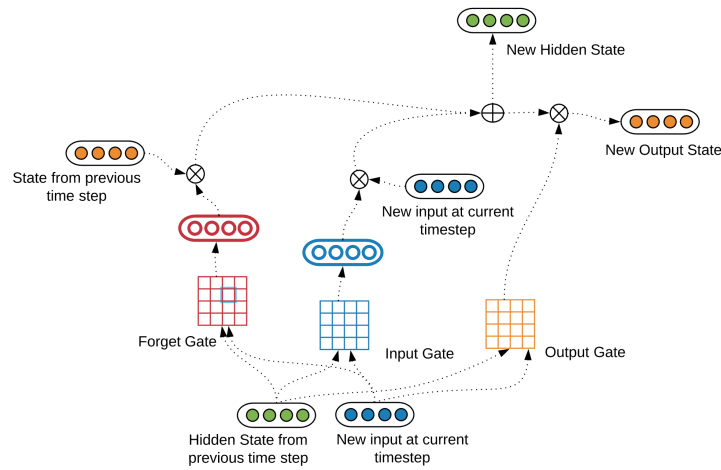
Two of the most widely used RNN cell implementation are Long Short-term Memory (LSTM) [HS97] and the Gated Representation Units (GRU) [Cho14]. In LSTM, there are three gates - input, forget and finally an output gate. The forget gate decides how much old information should be forget at the current time step, input gate decides what information from the latest word should be added into state vector, and finally combines these two to form a hidden state. In LSTM, a state vector are split into a hidden state and output state, where the latter is the former processed by output gate. The final output representation of the sequence from LSTM is based on the output state, which can be either the output state of last time step or averaged through all time step. For GRU (Figure 2.7), the major differences with LSTM is it does not have an output gate, thus the hidden state and output state is the same. Through experiments, GRU is proven to be effective as LSTM, but faster since fewer computation is needed [Chu14].

As mentioned, the major benefit of using the cells is it mitigates the gradient vanishing problem. While RNN uses the same back propagation tricks to train the model, RNN is essentially a multi-folded feed-forward neural network. In each fold of the network, a portion of the gradient will be consumed after passed through the activation function, which eventually becomes infinite small and stops the back propagation process. In LSTM and GRU cells, the gradients can pass through the cells without being changed due to the forget and reset gate. This is due to the forget and reset gate can be considered as the activation function for the cell, which is simply a set of weights and does not consume the gradients through normal activation functions like sigmoid functions.

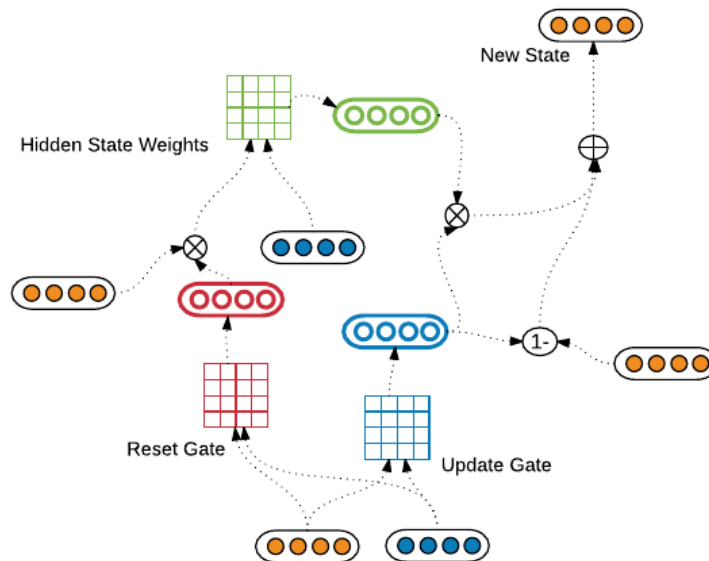
However, natural language sentence often contains non-sequential structure that is not ideal using a classic sequential model like RNN. For example, in idioms and phrases where the meaning of a word are considerably different from individual words, like 'blue' in 'out of the blue.' During these cases, it is not clear how sequential model should handle the words.

#### **2.2.4 Recursive Neural Network**

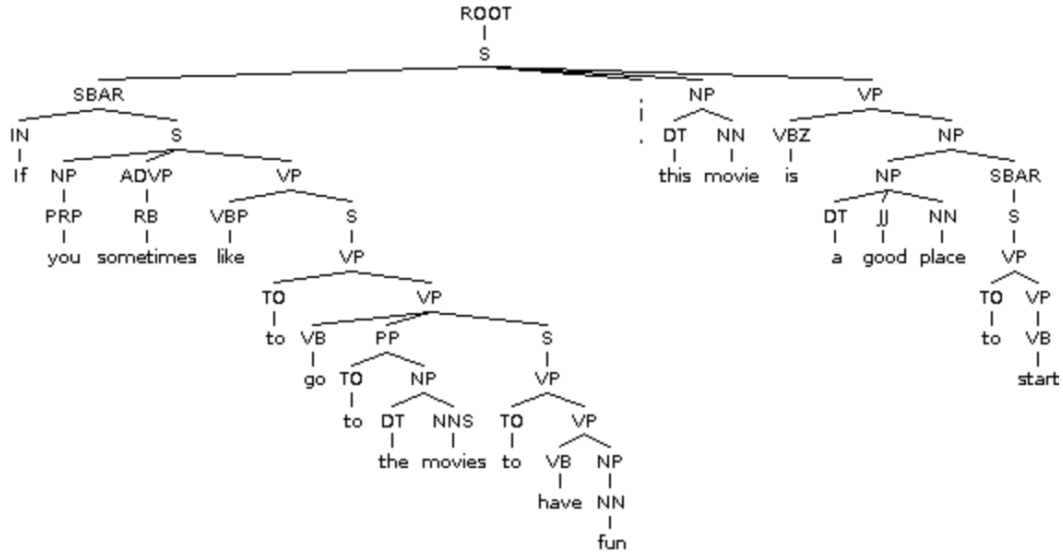
The last type of neural network is the Recursive Neural Net [Soc11]. While CNN handles neurons in geographical-like notion and RNN handle neurons in sequential order, the recursive neural network is built for tree structures. Neurons are connected only to their parent and children. All neurons share the identical weight matrices, similar to RNN, and compute representation following a bottom-up



**Figure 2.6** An example of LSTM cell. The cell contains a forget, input and output gate. Different from the most simple RNN model, the LSTM cell does not directly take the output state from previous time step, instead it pass along a hidden state in each time step, outputs are finally obtained by passing hidden states to output gates. The benefit of splitting output and hidden state it can detach output from the cells directly. Finally, the forget gate will learn to forget portion of previous hidden state, and input gate will learn to include a portion of the latest input



**Figure 2.7** An example of GRU Cell. GRU cell combines the output and input gate to update gate. While it associates output directly to the cells, due to the reset gate, it can still avoid gradient vanishing efficiently. The major benefit of GRU over LSTM is GRU can be computed faster due to fewer gates.



**Figure 2.8** An example of sentence parse tree with the example sentence “If you sometimes like to go to the movies to have fun, this movie is a good place to start”. We used Berkley parser in this example.

fashion. In the context of sentence, the tree structure is often related to the grammar structure of the sentence, which we provide an example in the following Figure 2.8:

In [Soc11], they first proceed the recursive neural network idea with simple neurons as in normal feed-forward neural network. In [Soc12b], in order to enhance the interaction between children, each node is attached to an additional matrix for semantic constitutionality (Figure 2.9). The matrix is used to model the children for each node, and the matrix will also propagate up in the tree to keep track of semantic development. In [Soc13], they further improve this concept by combining the word vector and word matrix into a tensor for richer representation.

Some extension of recursive neural network combines LSTM or CNN into the formula to improve learning. In [Tai15] and [Zhu15], they improved the model by replacing each node in the tree with a LSTM cell. In [Mou15]. they treated each (parent,left-child,right-child) pair in the tree as neighboring nodes and apply CNN filters onto it.

While being both fast and accurate, a major issue for recursive neural networks is the dependence of grammar structure. The output of the grammar parser directly influence how neurons in a recursive neural network are connected, and weak structure leads to wrong connection. Poor structure can derive from both the quality of the grammar parser and the quality of the sentence. Accordingly, comparing with other neural network models, recursive neural networks are subject to more preprocessing and filtering of the training data and has a more restricted use.

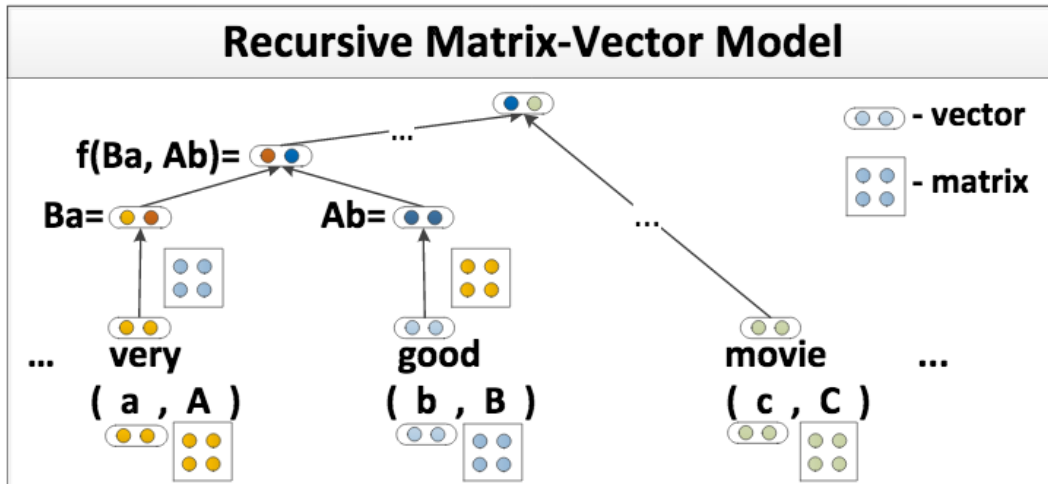


Figure 2.9 An example of Recursive Neural Network [Soc12b].

While each extension is built to solve a certain problem and are all proven to be very successful on sentiment classification comparing to traditional methods, each extension has certain drawbacks. The CNN models are in general bag-of-words models where word order are less concerned. RNN models, while the structure supports the cell be optimized for the whole sequence, it is potential to oversee important local structure of the sentence. Lastly, the performance of Recursive Neural Net is dependent on the quality of the tree structure and could fail if the sentence is not written following formal grammar structure. Our GRA model is built on top of these observations to provide a more flexible framework for sentiment classification. GRA combines a CNN and RNN to model linguistic information on both regional and global view, and did not utilize sentence grammar tree to better model unstructured cases.

## 2.3 Methodology

Figure 2.12 depicts an overview of the GRA model, which consists of three stages: the first stage generates phrase vectors using CNN; the second combines the word and phrase vectors, and incorporates word order to generate sentence representations through a soft-aligned RNN; the third stage makes class predictions based on these sentence representations. Figure 2.12 shows the processing flow for the  $i$ -th word, which is equivalent to the  $i$ -th time step.

### 2.3.1 Phrase Vector CNN

At the first stage in the GRA model, the model extract phrase vectors from word vectors using CNN. Each phrase vector is a representation of between two and five words (Figure 2.11).

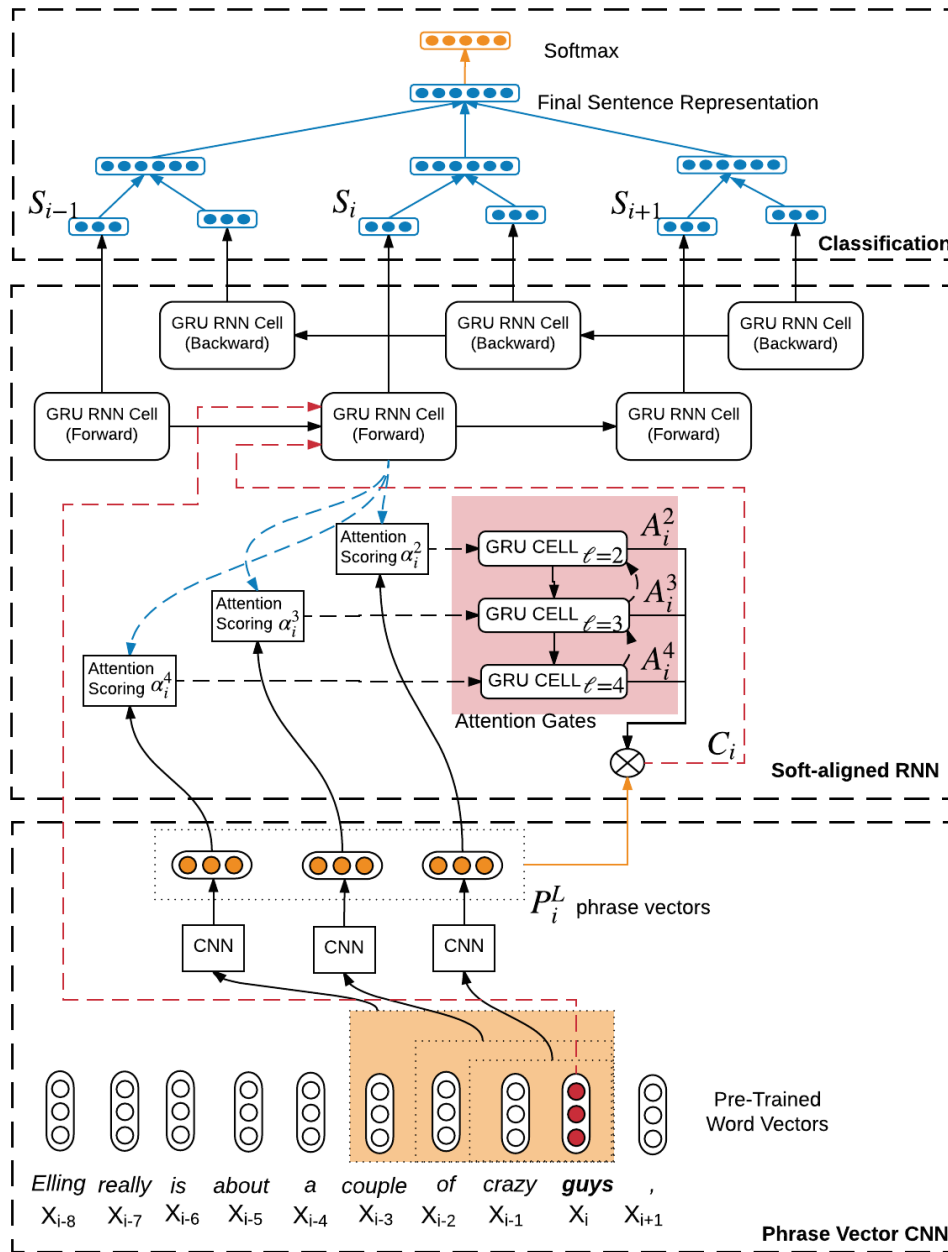
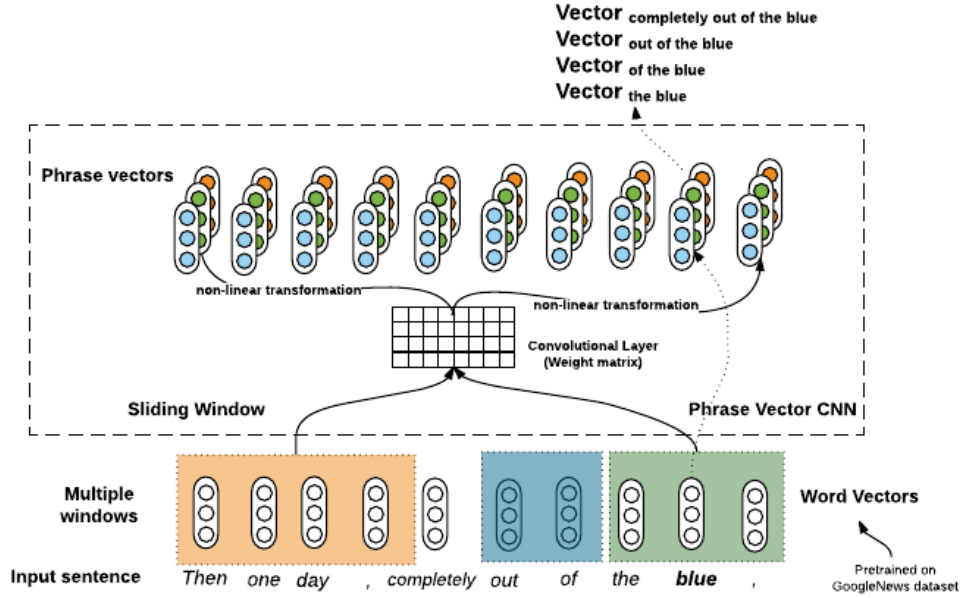


Figure 2.10 GRA Framework and Details at Step i.



**Figure 2.11** An example of CNN generating phrase vectors for word **blue**.

Let  $X_i \in \mathbb{R}^k$  represent a  $k$ -dimensional embedding for the  $i$ -th word in the sentence. An input sentence of length  $N$  can thus be considered as a vertical concatenation of  $X_{1:N}$ . A set of convolutional filters  $W_p^\ell$  and bias terms  $b_p^\ell$  is applied to the sentence as per equation (2.7), in order to learn a representation for each phrase of length  $\ell$ .

$$P_i^\ell = \text{Relu}(W_p^\ell \cdot [X_i, \dots, X_{i-\ell}] + b_p^\ell) \quad (2.7)$$

We use  $P_i^{L=\{2,3,\dots,\ell\}}$  to represent phrase vectors at time  $i$ , which includes all phrases ended with  $X_i$ .

### 2.3.2 Soft-aligned RNN

The second stage generates sentence vector representations (or states) using a soft-aligned RNN. The state updated with the  $i$ -th word is represented as a  $d$ -dimensional vector  $S_i$ .

The GRA model was inspired by an attention GRU-RNN model introduced by [Bah15], which was originally used for machine translation. The attention model provides an interface for a neural network to selectively include outputs from another model, which is ideal for the purpose of combining CNN and RNN.

For the  $i$ -th time step in GRU-RNN, the GRU cell forgets a portion of learned sentence information  $S_{i-1}$  using the update gate  $Z$ , and updates it through a reset gate  $R$ . In GRU cells, both gates are controlled by  $S_{i-1}$  and  $X_i$ . In GRA, another vector  $C_i$  combines the weight from the *Attention Gates* in Figure 2.7 with each phrase vector from CNN. This provides input to the GRU RNN cells, as

shown in equation (2.8).

An intuitive way to understand  $C_i$  is to consider that the model tries to determine which of the phrases generated by word  $X_i$  are more reasonable based on current sentence state  $S_i$ . In the example sentence shown in Figure 1, for the word ‘guys’, the weighting function determines weights for each of the phrase vectors representing ‘*couple of crazy guys*’, ‘*of crazy guys*’ and ‘*crazy guys*’ based on their similarity to the sentence state.

To compute similarity, both the phrase vectors  $P_i^\ell$  and the sentence state  $S_i^*$  are projected to a new vector space (after  $S_{i-1}$  is updated with  $X_i$ ), and then similarity is evaluated by a dot product, represented as  $\alpha_i^\ell$ . We call this step **attention scoring** and formalize in equation 2.9.

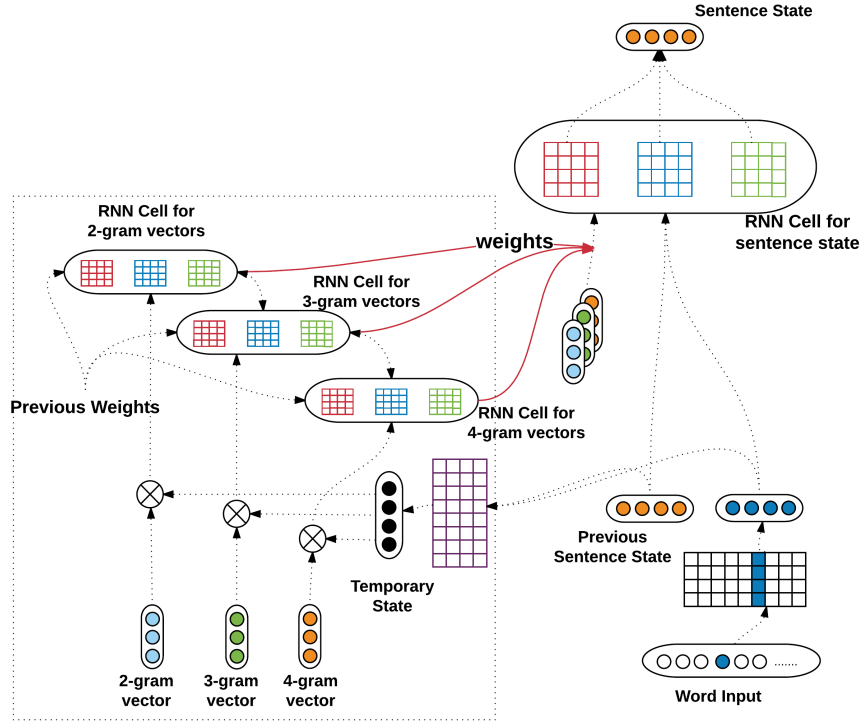
In[Bah15] attention framework, the underlying assumption was that one neural network always received the output of another. Applying softmax to the attention scores indicated that the receiving neural network must focus on a certain part of the input. However, this assumption might not hold in the GRA framework as phrase information is not always needed at each timestep of RNN training. For the example sentence “*Then one day, completely out of the blue, I had a letter from her.*”, it is clear that “blue” requires to be understood as a phrase (which is only meaningful as part of a phrase), but not for other words such as “I”. Accordingly, a loosely coupled framework that dynamically incorporates or omits phrase vectors is necessary.

The major challenge here is that the algorithm needs a reference to compute weights for the phrase vectors. For instance, in softmax, each input is simply weighted by its contribution to the sum. However, in GRA, the sum of similarity scores is not a good scaling factor since phrase vectors are sometimes omitted. Instead, we use a set of GRU cells that receive previous weights, other phrase’s weights, and attention scores as inputs, and use these to compute the final weights for each phrase vector. The intuition is that GRA is viewing weights as a sequence, and trying to determine the weight for  $P_i^\ell$  by concatenating attention scores, past weights and weights assigned to other phrase vectors. Using a RNN cell helps to store relevant past information and allows concurrent weights be easily added into the formula (Figure 2.12). To compute the weight for  $P_i^\ell$ , a GRU cell receives the weight for  $P_{i-1}^{\ell-1}$  if the weight for  $P_i^{\ell-1}$  is not computed yet. We called this process **attention gating**, and the final output is the set of weights  $A_i^\ell$  for the phrase vector  $P_i^\ell$ , as formalized in equation 2.10.

### 2.3.3 Classification Layer and Regularization

The penultimate layer of GRA, which outputs the final sentence vectors, averages sentence states from all time steps. Finally, classification is done using softmax to project the final sentence vector to  $K$  conditional probabilities, where  $K$  is the number of classes, and a class prediction is obtained from the **argmax** operation.

We implemented a bi-directional RNN with dropout for regularization [Pha14]. The RNN cells are shared for both forward and backward passes to limit the number of variables. This also helps to



**Figure 2.12** RNN cell in GRA. Each cell controls the attention scores for n-gram of different windows size. Each cells are connected, for example outputs from the cell for 2-gram will be used as inputs for the 3-gram cell.

decrease over-fitting.

**GRU RNN Cell**<sup>1,2,3</sup>:

$$\begin{aligned}
 Z_i &= \text{sigmoid}(W_Z \cdot [X_i, S_{i-1}, C_i] + b_Z) & H_i &= \text{tanh}(W_H \cdot [X_i, R_i \odot S_{i-1}, C_i] + b_H) \\
 R_i &= \text{sigmoid}(W_R \cdot [X_i, S_{i-1}, C_i] + b_R) & S_i &= (1 - Z_i) \odot S_{i-1} + Z_i \odot H_i
 \end{aligned}
 \tag{2.8}$$

**Attention Scoring:**

$$\begin{aligned}
 \alpha_i^l &= U_\alpha \cdot \text{tanh}((W_\alpha \cdot P_i^l) \odot S_i^*) + b_\alpha & \alpha_i^L &= [\alpha_i^2, \dots, \alpha_i^l] \\
 S_i^* &= W_s \cdot [S_{i-1}, X_i]
 \end{aligned}
 \tag{2.9}$$

<sup>1</sup>⊙ represents element-wise multiplication

<sup>2</sup>[A,B]represents horizontal concatenation of A and B

<sup>3</sup>W represents weight matrix used for the corresponding parameter, and b as bias terms

**Attention Gate**<sup>4,5</sup>:

$$\begin{aligned}
 AZ_i^\ell &= \tanh(W_{AZ}^\ell \cdot [\alpha_i^L, A_{latest}^{L-\ell}, A_{i-1}^\ell] + b_{AZ}) \\
 AR_i^\ell &= \tanh(W_{AR}^\ell \cdot [\alpha_i^L, A_{latest}^{L-\ell}, A_{i-1}^\ell] + b_{AR}) \\
 AH_i^\ell &= \tanh(W_{AH}^\ell \cdot [\alpha_i^L, A_{latest}^{L-\ell}, AR_i^\ell \odot A_{i-1}^\ell] + b_{AH}) \\
 A_i^\ell &= (1 - AZ_i^\ell) \odot A_{i-1}^\ell + AZ_i^\ell \odot AH_i^\ell \\
 C_i &= [Ai^2 \odot Pi^2, \dots, Ai^\ell \odot Pi^\ell]
 \end{aligned} \tag{2.10}$$

## 2.4 Datasets and Experimental Setup

The model is tested on datasets containing both ‘clean’ (i.e. well-structured) and ‘noisy’ text.

The clean datasets are obtained from **Stanford Sentiment Treebank (SST5)**, a 5-class movie review corpus (i.e. very negative, negative, neutral, positive, very positive) from Socher et al (2013). Labeling is done at both sentence and phrase level. Well-known sub-phrases (and individual words) are labelled separately for training, but are not used in testing. Dataset **SST2** is the same as SST5 but reduced to binary classes.

The noisy dataset is a 5-classes review dataset from **Yelp** [Tan15]. We parsed short reviews (less than 60 words) from the 200 most frequently reviewed restaurants. Also, we undersampled positive and very positive reviews as the reviews are skewed toward the positive end.

The accuracy results from the clean datasets were averaged over 5 runs using the train/test splits given in the datasets. The noisy dataset wasn’t broken down in this way in advance, so it was evaluated it using a 10-fold cross validation.

In order to minimize parameter tuning, we used the *Adadelta* [Zei12] optimizer to obviate the need to determine a learning rate. Dropout is set to 50% for each timestep in RNN, and no dropout in the penultimate layer.

During experiments, we set the dimension of word vectors to 300, and the CNN filter length to [2,3,4]. Each CNN filter has 150/50 dimensions in SST5,SST2/Yelp. Bi-directional RNN state size is set to 450/150 for SST5, SST2/Yelp for each direction. Each experiment lasts 10 epochs, with mini-batch size of 200. Similar to most benchmark models, GRA uses pre-trained word vectors<sup>6</sup> (trained on **GoogleNews**) to initialize the words embeddings. Words not present in the corpus are initialized randomly.

<sup>4</sup> $A_i^{L-\ell}$  represents  $\ell$  is excluded from L

<sup>5</sup>latest refers to  $i$  or  $i-1$ , depending if  $A_i^{L-\ell}$  is computed

<sup>6</sup><https://code.google.com/p/word2vec>

## 2.5 Results and Discussion

The classification accuracy of GRA and baseline methods are shown in Table 2.1. Results for baseline methods running against the SST5 / SST2 datasets are mostly taken directly from the corresponding papers<sup>7 8</sup>. For reimplemented baselines, we used the parameter settings specified in the original papers. It was only possible to run some of the baseline algorithms on the Yelp dataset due to availability of source code and parameter configurations. We explain each baseline methods in the following:

1. **LSTM and Bi-Directional LSTM.** This is the most basic recurrent neural network that is as described in previous sections. We used the exact same settings we have used in GRA in LSTM and Bi-Directional LSTM. The final sentence embedding is obtained by the average of all states in the model.
2. **DCNN: Dynamic Convolutional Neural Network** [Kal14a]. DCNN is a CNN application for sentence classification. In DCNN, the model attempts to create a multilayer framework that can represent different level of phrases in sentences. In each layer, the model will attempt to combine neighboring chunks through CNN, where each chunks are from a layer below or from the original word embeddings. At higher layers, the model can learn to associate phrases of longer distances. The height of the framework is dynamically adjusted to the length of the sentence.
3. **Paragraph-Vec** [LM14]. Paragarph-Vec is an extension of word vector models that adds a document or paragraph embedding to the training process. While training, each word will be associated with not only the context words but also the document embeddings. In other words, document embeddings can be seen as the weighted center of word embeddings of all words in the document. Classification is finally done on document vectors by another feed-forward network or SVM.
4. **CNN non-static and CNN static** [Kim14a]. This is an another attempt to use CNN on sentence classification. Different from DCNN where DCNN attempts to learn how to combine words through a hierarchical structure, CNN non-static uses a naive multiple window approach to scan through the sentence. Each window will be assign to different size, which can be seen as attempts to capture phrase of different length in the sentence. The difference between non-static and static is if the word embeddings are static in training or not.

---

<sup>7\*</sup> denotes that we reimplemented the algorithm, but reported SST5/SST2 results based on the results shown in their publications.

<sup>8</sup>Models without citation are implemented following parameter settings in section 3.

**Table 2.1** Accuracy of GRA and benchmarks. † denotes models that are trained on SST5 but sum the result of the softmax layer to obtain binary predictions; as stated in [Mou:2015], it is more difficult to obtain good results with this approach.

Methods	SST5	SST2	Yelp
LSTM (baseline)	46.4	85.9 <sup>†</sup>	56.5
Bi-Directional LSTM	49.5	86.1 <sup>†</sup>	57.8
DCNN [Kal14a]	48.5	86.8	-
Paragraph-Vec [LM14]	48.7	87.8	-
CNN non-static [Kim14a]	48.0	87.2	55.5
CNN multi-channel [Kim14a]	47.4	88.1	56.0
MG-CNN(w2v+Glv) [Zha16a]	48.2	87.9	55.8
MGNC-CNN(w2v+Syn+Glv) [Zha16a]	48.6	88.3	-
MVCNN [YS15]	49.6	<b>89.4</b>	-
GRA	<b>51.0</b>	87.9 <sup>†</sup>	<b>58.1</b>

5. **MG-CNN and MGNC-CNN: Multiple Group Norm Constraint CNN** [Zha16a] In these two models, they adopt the similar approach used in [Kim14a] but extended it by using multi-sources of word vectors. A more intuitive way to understand these two models is to consider word vectors learned from a locally-owned corpus and from Wikipedia is going to be different, and MG-CNN and MGNC-CNN attempts to build a more comprehensive sentence embeddings by considering a more diversified knowledge source. The difference between these two models is the how max-pooling is applied, while the first one applies a global max-pooling and the latter uses a local max-pooling for each knowledge source.
6. **MVCNN: Multichannel Variable-size CNN** [YS15] MVCNN can be seen as a hybrid of DCNN and MG-CNN, where the author uses both dynamic max-pooling size to combine words in a hierarchical structure, and also utilizes word vectors learned from multiple sources. They also introduced a pre-training method to avoid over-fitting by a CBOW-like model.

It can be seen from Table 2.1 that GRA outperforms the baselines on the fine-grained datasets (SST5 / Yelp), and is also comparable with the binary case (SST2).

Next, we further investigated the effect of soft-alignment, and compared GRA with structure dependent models for a more extensive analysis.

### 2.5.1 Effect of Soft-alignment

We first empirically evaluate the effect of soft-alignment by comparing GRA with/without soft-alignment on the **SST5** dataset. In the latter case, the last formula in formula set (4) becomes  $C_i = [P_i^2, \dots, P_i^\ell]$ , which can be seen as simply chaining together the two models. We added two more CNN and RNN hybrid models here for comprehensive comparison. Both hybrids combined CNN

**Table 2.2** Accuracy of GRA and other hybrids.

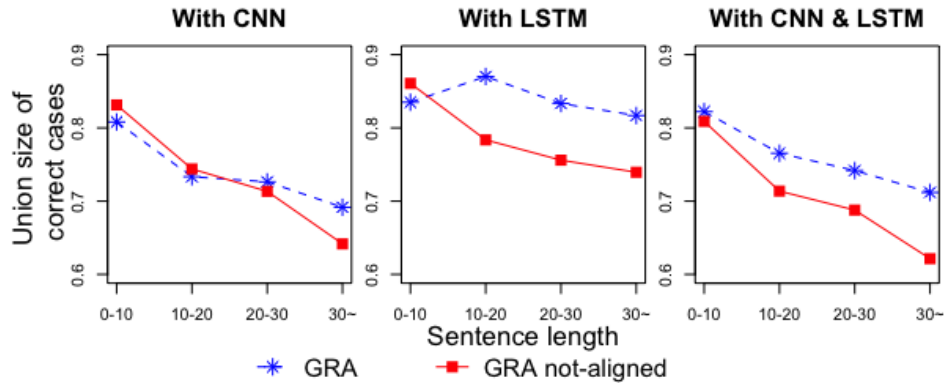
<b>Methods</b>	<b>SST5</b>
Average of softmax of CNN and RNN	50.2
Concatenate CNN and RNN	50.6
GRA not-aligned	48.8
GRA	<b>51.0</b>

and RNN at the penultimate layer, but the first one combined models by taking the average of the softmax scores; the second combined models by concatenating the sentence vectors generated by CNN and RNN. These two hybrid models can be seen as ensemble approaches since CNN and RNN are not interacting while generating the sentence vector. Results are shown in Table 2.2.

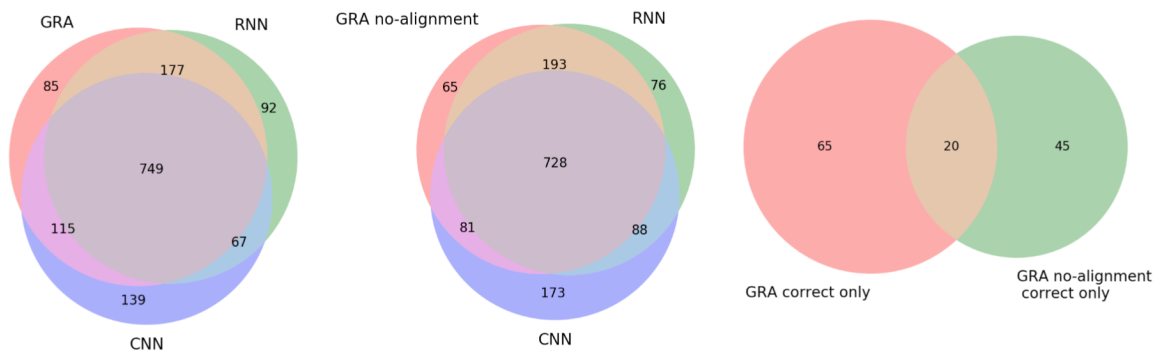
It can be seen from Table 2.2 that even very simple ensemble methods can yield good results when compared to standalone models. On the other hand, for GRA without alignment the result became worse when compared to RNN without phrase vectors (i.e. Bi-Directional LSTM in Table 2.1). We suppose that the drop of accuracy in the not-aligned version is a result of phrase vectors being over-counted with large weights, and thus reducing the effectiveness of the sequence learning ability in RNN. However, with soft-alignment, GRA can incorporate CNN phrase vectors into an RNN without impacting the sequence learning effectiveness.

We further investigated our assumption that GRA preserves more phrase level information without compromising the RNN part of the model. We evaluate this by quantifying the union of correct cases from GRA (both with and without soft-alignment) against the CNN/LSTM baselines. If soft-alignment helps to bridge the two models, then the predictions from GRA should be closer to those from CNN/LSTM with soft-alignment enabled than the not-aligned case. We show the results of this evaluation in Figure 2.13 using the test set from SST5. Each point shows the size of the union of correct cases for a variety of sentence lengths, and only for sentences that are predicted correctly more than 3 times in the 5 runs. When compared to LSTM and CNN/LSTM models, GRA with alignment produces a consistently larger union of correct cases (typically by 5-10%) than GRA without alignment. These results support the intuition that soft-alignment make an important difference.

We also explore the effect of the weight-alignment mechanism by examining how prediction results overlap with each other to complement our analysis. We examined the correctly predicted sentences of GRA and GRA not-aligned with CNN and RNN. In the following Figure 2.14 we showed the overlap of sentences that are correctly identified by either our GRA model, combination of CNN and RNN without weight alignment, RNN or CNN. One can see the GRA model has the larger overlap with CNN plus RNN comparing to GRA without alignment. GRA without alignment has a equal overlap with the RNN model with GRA with alignment, but does not cover as much in the CNN area.



**Figure 2.13** Coverage of CNN and LSTM correct cases between GRA and GRA-without-alignment.



**Figure 2.14** Figure on the left is the Venn diagram of sentences predicted correctly by GRA (our model) CNN, and RNN models. Figure on the middle is the same but using GRA without alignment. Finally, figure on the right is the Venn diagram considering only cases that are predicted correctly by GRA or GRA no-alignment. One can see that GRA outperforms the no-alignment version by covering more cases from CNN model and also utilize the combination of CNN and RNN to discover more sentences. Also, the two variations of GRA does not have much overlap.

Also, the GRA model has better synergy of CNN and RNN as it predicts more left-over sentences by CNN and RNN. The overlap results showed how weight-alignment help preserving more prediction power from CNN and RNN, and also creating better synergy of the two. An interesting discovery worthy to notice is shown in the right-most figure in Figure 2.14 that there are little overlap between GRA and GRA without-alignment.

In the following we showed a collections of sentences that are correctly or incorrectly predicted by only our GRA model (Table 2.3 2.4). By comparing the sentences that are correctly and incorrectly predicted by our GRA mode, we believe the GRA model is better at capturing tone changes in later part of the sentences. However, GRA tends to make more neutral prediction for sentences of shorter length. We believe it is the effect of more parameters in GRA which requires more training data to accurately capture short but strong sentiment sentences.

**Table 2.3** Example sentences that are correctly predicted by GRA only.

GRA / Label	Others	Sentence
Negative	Positive	It delivers some chills and sustained unease , but flounders in its quest for deeper meaning.
Very Positive	Positive	An epic of grandeur and scale that 's been decades gone from the popcorn pushing sound stages of Hollywood.
Neutral	Positive	This is a good movie in spurts , but when it doesn't work , it 's at important times.
Neutral	Negative	Bears resemblance to , and shares the weaknesses of too many recent action fantasy extravaganzas in which special effects overpower cogent story telling and visual clarity during the big action sequences.

**Table 2.4** Example sentences that are incorrectly predicted by GRA only.

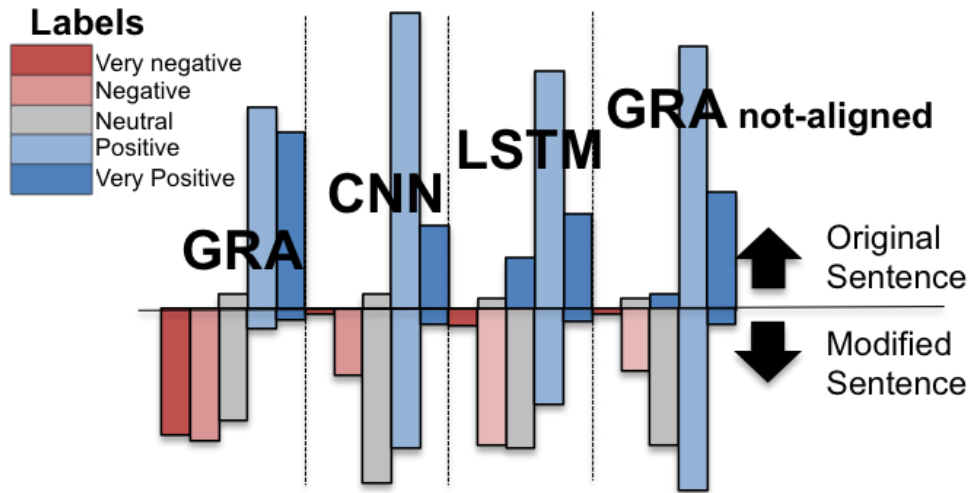
GRA	Others/label	Sentence
Neutral	Negative	Instead of accurately accounting a terrible true story , the film 's more determined to become the next Texas chainsaw massacre
Neutral	Positive	Moot point.
Negative	Very Negative	Murder and mayhem of this sort quickly becomes monotonous
Positive	Negative	Nearly surreal , dabbling in French , this is no simple movie , and you 'll be taking a risk if you choose to see it

Finally, we evaluated how sentimentally-sensitive the model is with soft-alignment by slightly modifying some of the sentences. We demonstrate in Figure 2.15 how predicted sentiments can be changed using a sample sentence. In Figure 2.15, We change the sentiment of sentence with minimal interruption, i.e. “good” to “not good” or “bad”. While all models reacted to the change significantly, GRA predicts a major sentiment shift and is the only one that changes the overall output prediction to negative. We believe the abrupt change in sentiment observed by GRA is caused by the model capturing phrase level changes.

## 2.5.2 Structure-dependent Models

In Table 2.5, we compare GRA with state-of-the-art structure-dependent models. Although we were only able to run one baseline against the noisy Yelp dataset (due to both availability of re-implementation and the lack of a good sentence-grammar tree), GRA shows comparable results to these models, and does no worse than second place for SST5 and SST2. Similarly, we explain each baseline in the following:

1. **MV-RNN: Matrix Vector Recursive Neural Network** [Soc12b]: MV-RNN is likely one of the first



**Figure 2.15** Change of sentiment distribution when sentiment of sentence is manually reversed. Sentiment distribution is obtained by feeding the derived sentence vectors to the softmax layer. The sample sentence was a **positive** sentence: *“If you sometimes like to go to the movies to have fun, this movie is a **good** place to start”*. I replaced *“a good”* with *“not a good”* to reverse the sentiment of the sentence.

experiment paper regarding recursive neural networks. In MV-RNN, each leaf node in the sentence syntax tree is initialized with a vector and a matrix. The vector contains the semantic meaning of each word, which is usually initialized with word vectors trained by distributed embedding models first. The matrix contains how each word should propagate the vectors to next layer, which can be seen as the controlling factor for the operation to combine semantic meanings for neighboring nodes. MV-RNN starts from the leaf node, and combine vectors of neighboring nodes to higher level nodes utilizing the matrices. This process will be carried out from the leaf nodes, and the vector of the final root node will then be used as the sentence representation.

2. **RNTN: Recursive Neural Tensor Network** [Soc13]: RNTN is proposed by the same authors for MV-RNN [Soc12b] later. While in MV-RNN each node will utilize a different  $d \times d$  matrix, where  $d$  equals to the size of word vectors, the number of parameters in the model can be overly large as it depends on the corpus size. In RNTN, they proposed a shared tensor of size  $d \times d \times d$  to decouple the controlling mechanism to corpus size while maintaining the ability to dynamically joining vectors of neighboring nodes.
3. **DRNN: Deep Recursive Neural Network** [IC14]: Comparing to MV-RNN and RNTN, DRNN also follows the bottom up strategy to combine vectors of child nodes to parent nodes. However, different from the other two, DRNN carries out the process using a shared **left** and **right** weight matrix to combine the left and right child node through out the sentence. Furthermore, by

**Table 2.5** Accuracy of GRA against structure dependent methods.† has same meaning as Table 2.1.

Methods	SST5	SST2	Yelp
MV-RNN[Soc12b]	44.4	82.9	-
RNTN[Soc13]	45.7	85.4	-
DRNN[IC14]	49.8	86.6 <sup>†</sup>	-
Dependency Tree-LSTM[Tai15]	48.4	85.7	55.2
Constituency Tree-LSTM [Tai15]	51.0	<b>88.0</b>	-
c-TBNN[Mou14]	50.4	86.8 <sup>†</sup>	-
d-TBNN[Mou14]	<b>51.4</b>	87.9 <sup>†</sup>	-
GRA	51.0	87.9	<b>58.1</b>

greatly reducing the size of parameters in Recursive neural nets, DRNN allows more layering options in the network. That is to say, each node in DRNN contains several layers, the vector of the node will thus be a combination of child nodes and vectors from underlying layers.

4. **Dependency and Constituent Tree-LSTM** [Tai15]: The Tree-LSTM model extends the idea of DRNN by including recurrent neural networks into recurrent networks. In Tree-LSTM models, the combination operation is controlled by a LSTM cell similar to recurrent models. Including LSTM cells into the recursive functions allows the combination operation to forget and update the nodes with greater flexibility. In the paper, the author experimented the Tree-LSTM idea on both dependency and constituent trees.
5. **c-TBNN and d-TBNN** [Mou15]: In these two models, the author combines recurrent networks and convolution networks. In TBNN models, they first compute the vectors for each node in the tree using other recurrent neural network models like MV-RNN, and then utilize a CNN convolutional filter on all parent, left child, right child triples.

## 2.6 Conclusion

We propose a novel structure-free method for combining RNN with CNN to improve sentence modeling. While CNN captures phrase-level information by convoluting sub-sentences, RNN preserves global sentence information. The soft-alignment mechanism helps to combine the two. Empirical results show that this hybrid model outperforms the baseline structure-free models, and performs similarly to structure-dependent models.

# RATIONALE-BASED RELATION EXTRACTION

## 3.1 Introduction

In this chapter, we will define the framework of rationale-based sentence level relation extraction. Sentence-level entity relation extraction is the task of recognizing the relationship between two entities within a sentence[Moe06; Sar08; Eic08; Ang15]. It is the core function of autonomous knowledge discovery in unstructured text (such as web documents), and thus critical for applications including information extraction, knowledge base construction, and many other higher level NLP tasks [Hen09; Niu12]. For example, given the following sentence:

*“He had chest pains and [headaches] $e_1$  from [mold] $e_2$  in the bedrooms.”*

with marked target entity  $e_1$  = “headaches” and  $e_2$  = “mold”, the goal would be to correctly identify that this sentence expresses a casual relationship from  $e_2$  to  $e_1$ , for which we use the notation Cause-Effect( $e_2, e_1$ ).

The field of entity-relation extraction has greatly benefited from recent developments in neural network models, and these benefits have propagated into many other important applications like document classification[Kim14a], sentiment analysis[Soc12b], summarization[See17], topic discov-

ery[Xu17], word-sense disambiguation[Liu15b], co-reference identification[CM16] and more[Hen09; Niu12]. Initially neural models pushed relation extraction performance to new heights, improving F1 score on relation extraction to 88% [Wan16] compared to 82.2% using an SVM with engineered features [RH10]. However, neural models often seem to perform incomprehensible operations on the sentence to reason desired knowledge, whereas human readers usually derive relations between entities by locating key indicating words. For instance, given the following sentence, a human will likely focus only on a few keywords that explain the core relationships, such as the Component-Whole( $e_1, e_2$ ) relationship, whereas a neural model will leverage every word in the sentence based on computed scores:

**Example sentence:** *"The disgusting scene was retaliation against her brother Philip who rents the [room] $e_1$  inside this apartment [house] $e_2$  on Lombard street."*

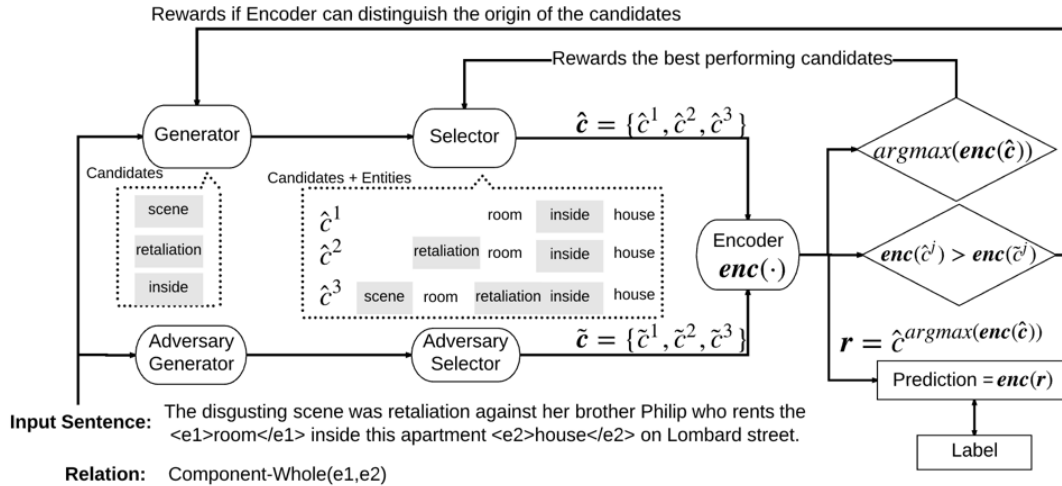
**Human interpretation:** *"The disgusting scene was retaliation against her brother Philip who rents the [room] $e_1$  **inside** this apartment [house] $e_2$  ~~on~~ Lombard street."*

**Attention-based CNN models:** *"The<sub>0.02</sub> disgusting<sub>0.03</sub> scene<sub>0.1</sub> was<sub>0.02</sub> retaliation<sub>0.03</sub> against<sub>0.07</sub> her<sub>0.015</sub> brother<sub>0.02</sub> Philip<sub>0.04</sub> who<sub>0.08</sub> rents<sub>0.03</sub> the<sub>0.1</sub> [room] $e_1$  inside<sub>0.13</sub> this<sub>0.03</sub> apartment<sub>0.09</sub> [house] $e_2$  on<sub>0.01</sub> Lombard<sub>0.03</sub> street<sub>0.08</sub>."*

**(value behind the words stands for the derived importance of the word for determining the relation)**

In this research, we aim to improve interpretability of neural models by imitating the logic used by human annotators through a three-stage Generator-Selector-Encoder approach. The Generator first enumerates candidate text fragments from the sentence using a Recurrent Neural Network (RNN) model, then the Selector extracts fragments that are informative for determining the relation and passes these on to the Encoder to make final relation predictions. We refer to the text fragments as **rationales** as per the definition in [Zai07; Zha16b; Lei16]. Some good examples of rationales are like "inside" in the above example sentence, or "caused", "resulted in" and "leads to" for entities of Cause-Effect relationship.

However, under most situations, a heavily annotated dataset with rationales is often too expensive to create; hence our Generator and Selector are mostly trained to identify rationales in a unsupervised manner. The Generator and Selector employ adversarial reinforcement learning where they are rewarded when their outputs are favored in the Encoder over a separate adversary set of Generator and Selector. We illustrate the process in Figure 3.2. Finally, as many learning systems can be manually optimized, we facilitate a refinement process in our pipeline. The refinement process provides an interface for users to examine and correct rationales to refine the predictions. After refinement, the model is then retrained to incorporate the changes. When refinements are provided, the Generator and Selector are trained in a semi-supervised manner.



**Figure 3.1** Model Structure with sample input sentence. Entities are given from the dataset. The Generator selects candidate rationales, and the Selector enumerates all possible combinations of candidates with entities and selects one ‘best performing candidate set’. An adversary Generator and Selector carry out the same process using a randomized approach. Candidate sets are then transformed into a vector representation by the Encoder and are evaluated against the ground-truth. Rewards are fed back to the Generator if the Encoder is able to identify candidate sets that are generated randomly, while the Selector is rewarded if the best performing rationale candidate set outperforms the one in the adversary. The best performing candidate set from Selector are finally considered as the extracted rationales, and then used in the Encoder to predict entity relation. In this example, the Generator first selected “scene”, “retaliation”, “inside” as candidates, then the Selector generates phrases using the entities - room and house, and the candidates. After the phrases are generated, it passes to the Encoder to be scored, and the best scored set is rewarded in the Selector, which in this case should be the phrase “room inside house”. Meanwhile, the Adversary Generator might randomly generate phrases by randomly selecting candidates, the Generator will be rewarded if it performs better than the Adversary Generator

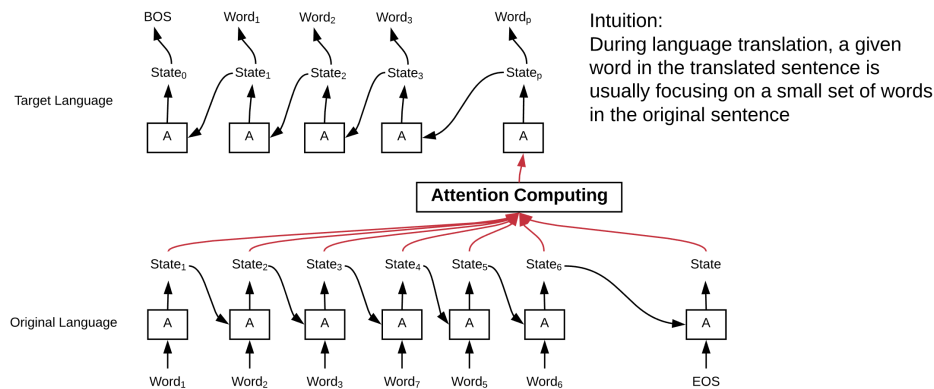
## 3.2 Interpretable Networks and Generative Networks

### 3.2.1 Interpretable neural network

Interpretable neural network models have recently become of increased interest to the neural network community [Dum09; Sim13; Ngu16; Mon17]. While the general goal is to obtain justifiable results or provide leads to visualize the process inside neural networks, we can roughly split these models into two categories by the degree of engineering of the inputs.

We begin with the type of models that are less inclined to modify the inputs. The most known approach is Activation Maximization (AM) where the goal is to discover the characteristics of inputs that excites certain neurons the most. In [Dum09], they first proposed a gradient ascent approach to for AM that attempts to find the most salient sample from the data set that activates certain neurons. The gradient ascent approach is done after training and fixing the network parameters, but maximize gradients by make inputs as the factor to be trained. In the same paper, they proposed another method that relies on permuting neuron weights in a Deep Belief Net(DBN). A DBN is a generative network that stacks multiple generative layers, and while each  $i_{th}$  layers attempts to create  $i + 1_{th}$  layer, each  $i + 1_{th}$  is mapped backward to  $i_{th}$  layer to reproduce the results. Another way to see DBN is as a compress-decompress network. In [Dum09], they proposed one can set weight for certain neurons to 1 and the others 0 after training, which in effect disentangles the effect of each input to the final results. Later, Simonyan et al. [Sim13] extended the idea used in [Dum09] to supervised models, where each neuron can be directly connected to the labels while in [Dum09] it requires specifically annotated inputs to explain the contribution of each neurons. In [Ngu16], they leveraged several different image generating models to create a synthesized image for each class to further disentangle the effect of each element in the input. For example, if a neuron seems to be activated by a input that contains  $A$  and  $B$ , [Ngu16] will try to maximize neurons by generated inputs that contains only  $A$  or  $B$ . Finally, while many AM methods require human evaluation to assess interpretability, [Sam16] provided evaluation methods and metrics for less biased assessment.

In the second type of models, inputs are usually discriminated before fed to neurons, and the degree of discrimination provides materials for interpretability. The attention model from [bahdanau2014] is one of the most influential idea in this approach. In [bahdanau2014], their goal is to perform language translation through two neural networks: a encoder that encodes the original text into a vector and a decoder decodes the vector to text in target language. They proposed that by adding an attention mapping for words in the original and translated text, it can help the decoder to put more emphasis on more related words in the original text. Several applications utilizing the attention idea have then be proposed in many domains including relation extraction [Rus15; Yan16; Wan16] since attention is easier to compute and provides a quick overview of highlights in the inputs relates to the goal. Our GRA model explained in the previous chapter is also an extension of the



**Figure 3.2** Attention model used in [bahdanau2014]. The model was originally used for language translation. Prior [bahdanau2014], language translation used a Encoder-Decoder approach that combines two Recurrent neural networks, where states for each word in the Encoder will be feed to the Decoder then attempts to generate corresponding translated word, a major drawback is the Encoder and Decoder is not aligned. That is, the Decoder will naively try to create a word that matches to every word in the original language. Attention model is introduced to weight outputs from Encoder prior pass to the Decoder, words with higher attention resembles the which words the Decoder should focus on. While it is used initially in language translation, it is then used in many different areas as it is efficient and simple.

attention models.

Another type of model that also falls into the second type is called rationale-based models [Zai07; Lei16; Wu17]. While similar to attention models where the model attempts to discriminate inputs prior inputting to neurons, rationale-based models take a more aggressive approach by assigning only attention of 1 or 0 to the inputs. That is to say, only a portion of inputs will be used in the rationale-based models. Inputs that are not masked are called rationales, and the idea for rationales is that it should be most directly related to the target goal then other inputs in the data entry. The criteria for selection can be totally data-oriented or manually curated. The ultimate perk of posing constraints on the model is the proximity to human reasoning which in turn gives better interpretability.

Our work is essentially in the rationale based models as suggested in the title, and is the first to explore rationales in relation extractions. There are two major difference comparing to other rationale models besides the type of task. First, in [Zha16b], rationales assumed given and the goal is to find how to utilize the rationales for better classification. In our work, rationales needs to be learned from the data. This allows our model to be easier to generalize to relation extraction for different domain areas. Secondly, comparing to [Lei16] where rationales are well-diverse, rationales in our work are highly overlapped. This is due to the goal in [Lei16] is aspect level sentiment analysis for beer reviews, and rationales for different aspect of reviews - taste, smell, and palate are understandably not similar. However, rationales for relation extraction are often shared between different

relations, like the word “of” or “by” is commonly used for many different relations (Figure ??). In effect, while the goal of [Lei16] is to select a short segment of text that can replace the original text for different review aspects, and the goal of our research is to find rationales that can better support relation extraction that other words in the sentence. Due to the difference, in [Lei16] they select rationales through Generative Adversarial Networks, and we select rationals through a combination of Generative Adversarial Networks and Noise Contrastive Estimation which will be explained in the following.

<p>EXAMPLE 1. a beer that is not sold in my neck of the woods , but managed to get while on a roadtrip . poured into an imperial pint glass with a <b>generous head that sustained life throughout</b> . nothing out of the ordinary here , but a good brew still . body <b>was kind of heavy , but not thick</b> . the hop smell was excellent and enicing . very drinkable</p> <p>EXAMPLE 2: a : poured a <b>nice dark brown with a tan colored head about half an inch thick , nice red/garnet accents when held to the light . little clumps of lacing all around the glass</b> , not too shabby . not terribly impressive though s : <b>smells like a more guinness-y guinness really</b> , there are some roasted malts there , signature guinness smells , less burnt though , a little bit of chocolate ... .. m : relatively thick , it is n't an export stout or imperial stout , but still is pretty heavy in the mouth , <b>very smooth , not much carbonation . not too shabby</b> d : not quite as drinkable as the draught , but still not too bad . i could easily see drinking a few of these .</p>	<p><b>OF</b>  Component-Whole: &lt;e1&gt;configuration&lt;/e1&gt; <b>of</b> .... &lt;e2&gt;elements&lt;/e2&gt;  Member-Collection: &lt;e1&gt;crowd&lt;/e1&gt; <b>of</b> &lt;e2&gt;candidates&lt;/e2&gt;.  Message-Topic: &lt;e1&gt; speech was summary <b>of</b> &lt;e2&gt; ... problems</p> <p><b>In</b>  Entity-Destination:&lt;e1&gt;guns&lt;/e1&gt; ... <b>in</b> ...&lt;e2&gt;safe&lt;/e2&gt;  Member-Collection:&lt;e1&gt;sergeant&lt;/e1&gt; <b>in</b> &lt;e2&gt;army&lt;/e2&gt;</p> <p><b>BY</b>  Caause-Effect: &lt;e1&gt;dips&lt;/e1&gt; caused <b>by</b> ... &lt;e2&gt;y-rays &lt;/e2&gt;  Product-Producer:&lt;e1&gt;firm&lt;/e1&gt; co-founded <b>by</b> ... &lt;e2&gt;head &lt;/e2&gt;</p>
--	--

**Figure 3.3** Comparison of rationales between [Lei16] and this work. Examples of [Lei16] is of the left, where each color indicates rationales of different aspect. Examples of rationale in this work in on the right, where each example begins with the relation type, words without entity tags are selected rationale from the model. Words in bold indicate overlapping rationale in different relation type.

### 3.2.2 Generative Adversarial Network and Noise Contrastive Estimation

Most machine learning algorithms are designed to fit some probability distributions to the observations, and involves fitting many parameters  $\theta$  in the model. While one of the most direct method to find the optimum  $\theta$  is Maximum likelihood estimation (MLE) , for some cases the MLE solution is intractable, and both GAN and NCE are both methods to approximate solutions for intractable MLE cases. In other words, both GAN and NCE produces some generated outputs to fit MLE.

Both GAN and NCE are engineered toward to employ a binary classifier to identify a given input  $x$  is drawn from the real data (denoted as  $y = 1$ ) or not(denoted as  $y = 0$ ), and are both optimizing against a distinguishability game value function :

$$V(p_c, p_g) = \mathbb{E}_{x \sim p_d} \log p_c(y = 1|x) + \mathbb{E}_{x \sim p_g} \log p_c(y = 0|x) \quad (3.1)$$

where  $p_c(y = 1|x)$  represents that  $x$  comes from the true input data distribution  $p_g$ , instead of from a fake distribution  $p_g$  that is generated by a fake, imposing generator.

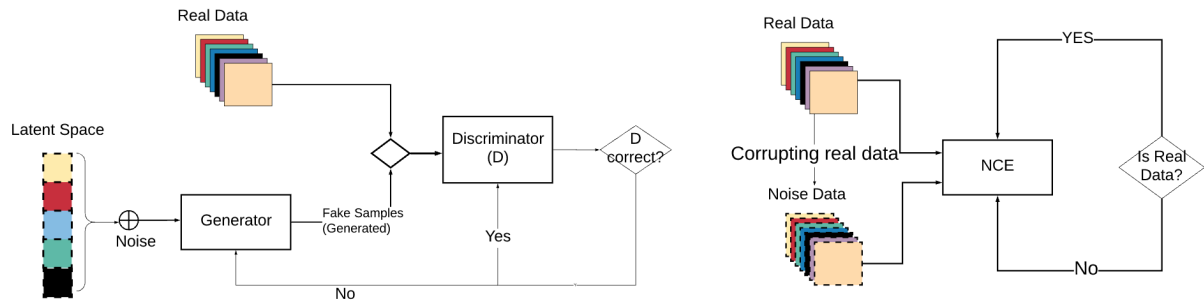
In NCE, the model used a fixed noise distribution for  $p_g$ , and the goal of NE is to learn a model  $p_m(x)$  for the following:

$$p_c(y = 1|x) = \frac{p_m(x)}{p_m(x) + p_g(x)} \quad (3.2)$$

NCE is commonly used in distributed embedding models like training word vectors, as it provides an alternative to exact classifications which speeds up the process.

In GAN, there does not exist  $p_m$  and the goal is to  $p_g$ . Also, instead of treating  $V$  as an optimization function to optimize against, it defines a minimax game on  $V$  composed by a Generator and Discriminator. A common way to understand this is Generator attempts to optimize  $p_g$  to make it closer to  $p_d$  and able to trick the Discriminator.

While the two model shares certain similarity in method and in goal, the major difference exist in the training periods. In GAN, the Generator and Discriminator and trained in a two-step fashion, just like Expectation-Maximization(EM) algorithm. This leads to two local optimums of  $V$  where the first is the optimum for Generator and the second is for Discriminator. This is different from NCE while in NCE it optimize the whole model in every training iteration, leading to global optimum for  $V$ . (Figure3.4)



**Figure 3.4** Comparison between GAN and NCE. The GAN model is on the left and NCE on the right. In GAN, there are two optimizable components: Generator and Discriminator. In NCE, there's only a NCE model to optimize. GAN will optimize Generator and Discriminator separately, and NCE will optimize the model using real data and noise data the same time.

We posit our work in between NCE and GAN. For one, our model is close to NCE as the goal of our model is in essence a MLE model for classifying the sentences to different relation types. As for GAN, we utilize the Generator and Discriminator idea in our model to extract rationales from the Generator. As a result, we adopted the two-step training in GAN, and our model is not guaranteed to converge at global optimum. We explain how we combine the two models for rationale generation

in next section.

### 3.3 Adversarial Rationale Generation

We formalize here the general idea of rationale generation for entity relationship extraction in this research. Consider an input sequence of words  $\mathbf{x} = \{x_t\}_{t=1}^T$ , where  $e_1$  and  $e_2$  are the entities of interest and  $e_1$  and  $e_2 \subset \{x_1, x_2, \dots, x_T\}$ .  $T$  is the length of the input sequence. The goal of relation extraction is to use  $(\mathbf{x}, e_1, e_2)$  to predict the relationship  $\mathbf{y}$  between  $e_1$  and  $e_2$ . On top of the representation learning regime this research falls into, we refer to the prediction process as encoding the matrix  $(\mathbf{x}, e_1, e_2)$  to a target vector that is representative of the desired relationship. We use  $enc(\mathbf{x}, e_1, e_2)$  to represent this process. The challenge in this research lies in the complexity within the encoding structure that obscures interpretability. This results in two problems: 1) unjustifiable results, and 2) difficulty in adjusting the model when errors are detected.

The general idea of rationale generation for relation extraction is to extract a set of rationales  $\mathbf{r}$  that are incorporated into the encoding process as  $enc(\mathbf{r}, \mathbf{x}, e_1, e_2)$ , where  $\mathbf{r} \subset \{x_1, x_2, \dots, x_T\}$  but  $\mathbf{r} \notin \{e_1, e_2\}$ . The goal of selecting rationales is to provide a set of terms that help to justify relationships in the sentence, and that make sense to human readers. For example, consider the following sentence:

**Example sentence:** “*The disgusting scene was retaliation against her brother Philip who rents the [room] $e_1$  inside this apartment [house] $e_2$  on Lombard street.*”

This entails a Component-Whole( $e_1, e_2$ ) relationship, hence good rationales should be words like  $\{inside, apartment\}$  which readers would use to describe the relationship.

We split the rationale generation process into two steps: the first step is candidate generation  $gen(\mathbf{x}, e_1, e_2)$ , which samples  $\mathbf{c} = \{c_t\}_{t=1}^T$  from the input and  $c_t \in [0, 1]$  represents that whether  $x_t$  should be considered as a rationale ; the second step is to sample rationales  $\mathbf{r}$  from  $\mathbf{c}$  and is represented as  $sel(\mathbf{c}, e_1, e_2)$ .

We can consider this as an adversarial problem where our target is not to find the best words that the Encoder can reason upon, but instead to find a set of words that is more useful for the Encoder than another set of words. Hence, the goal of the Generator and Selector is to generate rationales that  $enc(\mathbf{r}, \mathbf{x}, e_1, e_2)$  can outperforms another set of rationales  $enc(\tilde{\mathbf{r}}, \mathbf{x}, e_1, e_2)$ .

Building on the adversarial approach, the selection of rationales can be completely unsupervised. However, another desirable characteristic is to make the model capable of human intervention. While the rationales are trained unsupervised, we introduce a  $\mathbf{r}_{re}$  factor where “re” indicates refinement, and can be used to support semi-supervised training.

A difference between this model and previous work is worth noting: in [Lei16], the aim is to generate concise and reasonable summaries for customer reviews. In their research, the goal of

$gen(x)$  is to generate rationales  $r$  that ensure that  $enc(x)$  and  $enc(r)$  are at similar points in the target vector space. In other words, the goal of  $r$  in their research is to serve as a proxy of  $x$  and fool  $enc(\cdot)$ , while in our work the goal of  $r$  is to outperform all possible short enumerations of  $x$  in  $enc(\cdot)$ .

### 3.4 Encoder, Generator and Selector

We now describe the details of each component in our model. We begin by describing the Encoder in order to highlight the key contributions of our model.

#### 3.4.1 Encoder

Given an input training tuple  $(x, e_1, e_2, y)$ , where  $x = \{x_t\}_{t=1}^T$ , and  $y$  is the one-hot  $m$  dimensional vector representing the relation class. The goal of the Encoder  $enc(r, x, e_1, e_2)$  is to produce a probability distribution  $\hat{y}$  that approximates to  $y$ , given the set of rationales  $r$  as explained in the previous section. The quality of the Encoder is thus determined by the closeness between  $y$  and  $\hat{y}$ , where closeness is often referred to as loss, and quantified as:

$$loss = \sum_{i=1}^{|N|} -y_i \cdot \log(\hat{y}_i) \quad (3.3)$$

where  $N$  is the training set and  $|N|$  is the size of the set.

The Encoder performs two computations: 1) it generates a representation of  $(r, x, e_1, e_2)$ , and 2) it generates  $\hat{y}$  by projecting the representation of  $(r, x, e_1, e_2)$  to the  $y$  space. The second part is achieved by a simple feed-forward neural network with a softmax function. The first computation consists of two parts: computing  $V_x$  which represents the encoded sentence, and  $V_r$  which represents the encoded rationales  $r$  and the entities  $(e_1, e_2)$ .

To compute  $V_x$ , we incorporate a Convolutional Neural Network (CNN) model, which aims to represent the overall sentence meaning. The choice of CNN is based on recent successes in NLP applications [Kim14b; Kal14b]. As in [Kim14b], the CNN model contains a set of convolutional filters  $W^\ell$  and bias terms  $b^\ell$ . These filters are applied to all possible consecutive n-grams of length  $\ell$  in the sentence so that we can learn a representation for every n-gram in the sentence. Finally, to derive a sentence representation from n-gram representations, we adopt the commonly used approach of taking only the most significant feature for each dimension of the n-gram representations - this is known as *max-pooling*.

We compute  $V_r$  through a feed-forward neural network with a weight matrix  $W_r$  and a bias term  $b_r$ . Note that the input is order sensitive. For example, if  $r = \{x_{10}, x_{15}\}$ ,  $e_1 = x_4$  and  $e_2 = x_7$ , which

means the model uses the 10th and 15th word of the sentence as rationale, the input should be sorted as  $(e_1, e_2, x_{10}, x_{15})$ , whereas the input will be  $(x_3, e_1, e_2, x_{10})$  if  $r = \{x_3, x_{10}\}$ .

In summary, the Encoder  $enc(r, x, e_1, e_2)$  receives as input the selected rationales, the entities, and the target sentence. In mathematical terms:

$$\begin{aligned}
 \hat{y} &= enc(r, x, e_1, e_2) \\
 &= softmax(W_{enc} \cdot [V_r \oplus V_x \oplus e_1 \oplus e_2] + b_{enc}) \\
 &\quad (\oplus = \text{vector concatenation})
 \end{aligned}
 \qquad
 \begin{aligned}
 V_r &= Relu(W_r \cdot [r, e_1, e_2] + b_r) \\
 V_{x_t}^\ell &= Relu(W_x^\ell \cdot x_{t:t+\ell} + b_x^\ell) \\
 V_x^\ell &= [V_{x_1}^\ell, V_{x_2}^\ell, \dots, V_{x_{T-\ell}}^\ell] \\
 V_x &= \{maxpooling(V_x^\ell)\}^{\ell \in L}
 \end{aligned}$$

### 3.4.2 Generator

The goal of the Generator  $gen(x)$  is to extract pieces of text from input sentence  $x$  that can be used to connect and predict the relationships between target entities in the sentence. A different way to view the role of the generator is to derive another set of binary variables  $c = \{c_t\}_{t=1}^T$  where each  $c_t \in [0, 1]$  is whether  $x_t$  is chosen as a candidate rationale. We represent it as  $c \sim gen(x)$ .

We adopted a recurrent neural network (RNN) model in this work to compute  $c$  due to the sequential characteristics of  $x$  and  $c$ . RNNs are ideal for sequential modeling, where the output of step  $t$  is used as a part of the input for step  $t + 1$ . Specifically, all inputs of the model share the same set of transformations and output functions, where these functions help to project inputs to the target dimensional space, and are fed back for use in the next step. In this work, we use a bi-directional RNN where the input is modeled from both ends, and shares the same parameters in both directions. In the formal description in Equation (3), both  $W$  and  $U$  are weight matrices, and  $b$  are bias terms.

Recurrent Unit :

$$\begin{aligned}
 z_t &= \sigma(W_z \cdot x_t + U_z \cdot h_{t-1} + b_z) & \overleftarrow{h}_t &= RecurrentUnit(x_t, \overleftarrow{h}_{t+1}) \\
 o_t &= \sigma(W_r \cdot x_t + U_r \cdot h_{t-1} + b_r) & \overrightarrow{h}_t &= RecurrentUnit(x_t, \overrightarrow{h}_{t-1}) \\
 \hat{h}_t &= \tanh(W_h \cdot x_t + U_h \cdot (h_{t-1} \circ o_t) + b_h) & s_t &= sigmoid(W_s \cdot [\overleftarrow{h}_t, \overrightarrow{h}_t, e_1, e_2] + b_s) \\
 h_t &= z_t \circ \hat{h}_t + (1 - z_t) \circ h_{t-1} \\
 &\quad (\circ = \text{elementwise-multiplication})
 \end{aligned}$$

In the most simplistic generator, the probability that  $x_t$  is chosen is conditionally independent of all other  $x$ . In other words, we can sample candidate  $c$  from a uniform distribution using  $\{s_t\}_{t=1}^T$ . However, we observed from the data set that our target rationales often consist only very limited

words - for example “*room **inside** apartment*”. We thus added the following Equation (4) for  $c$  to limit the number of rationales selected to be at most  $J$ .

$$c_t = \begin{cases} 1, \text{sort}(\{(s_t > \text{rand}(0, 1)) \cdot s_t\}_{t=1}^T)[1:J] \\ 0, \text{otherwise} \end{cases} \quad (3.4)$$

$$c = \{c_t\}_{t=1}^T$$

### 3.4.3 Selector

The last component of the model is the Selector  $sel(c)$ , where the goal of the Selector is to choose rationales  $r$  from  $c$ . The Selector is a feed-forward neural network can be considered as  $r \sim sel(c)$ , and defined as follows:

$$\hat{c}_t^j = \begin{cases} 1, \text{sort}(\{c_t \cdot s_t\}_{t=1}^T)[1:j], j \in [1:J] \\ 0, \text{otherwise} \end{cases}$$

$$\hat{c}^j = \{\hat{c}_t^j\}_{t=1}^T, \text{ where } \hat{c}_t^j = 1$$

$$score(\hat{c}^j) = W_c \cdot enc(\hat{c}^j, x, e1, e2)$$

$$scores^j = \text{softmax}(\{score(\hat{c}^j)\}_{j=1}^J)^j$$

$$sc\hat{o}res^j = \begin{cases} 1, j = \text{argmax}(scores) \\ 0, \text{otherwise} \end{cases}$$

$$r = \{x_t\}, \text{ where } \hat{c}_t^j = 1, j = \text{argmax}(scores) \quad (3.5)$$

Notice that, in order to generate scores for each candidate, the model adopts the same weight functions in the Encoder section. A more intuitive way to understand this is that the Selector scores each candidate by their expected projection and contribution to the  $y$ -space. For instance, if using two rationales ( $j = 2$ ) outperforms using only one rationale ( $j = 1$ ), the Selector will choose to use two rationales.

### 3.4.4 Joint Objective and Adversarial Training

In order to jointly train the three components, we formulate a joint loss function that binds the components together. However, the major challenge lies in the fact that both the Generator and Selector are not provided with ground-truth labels, and hence are unsupervised. In order to facilitate this model, we formulate an adversarial training regime.

Our adversarial training model is similar to that used in the GAN model [Goo14]. In previous work, the example application was generating images based on sentences, where the Generator generates a fake image that tries to fool the Encoder. After training, the Generator becomes effective at generating fooling images, and the Encoder become discerning at identifying these fakes. We

applied this same concept in our model but with one key modification: all the components of our model work together and compete with an adversary Generator and Selector.

In our model, the Generator competes with an adversary Generator, for which we use a randomized approach. The adversary Generator will sample  $\tilde{c}$  randomly from  $x$ . The adversary rationales  $\tilde{r}$  will then be selected using  $\tilde{c}$  as per Equation (5).  $\tilde{r}$  will be passed to the Encoder to generate a projection onto the target dimension, which we consider as  $\tilde{y}$ . We compare  $\tilde{y}$  with  $\hat{y}$  in terms of their similarity to  $y$  and is denoted by  $\mathcal{D} \in \{0, 1\}$ .  $\mathcal{D} = 1$  when  $\hat{y}$  is closer to  $y$  comparing to  $\tilde{y}$  and is noted as ‘model success case’, meanwhile  $1 - \mathcal{D} = 1$  for the opposite situation and is noted as ‘adversary success case’. We further split  $\hat{y}$ ,  $\tilde{y}$  and  $\mathcal{D}$  into  $\hat{y}^j$ ,  $\tilde{y}^j$ ,  $\mathcal{D}^j$  and  $j \in [1 : J]$ , where  $\mathcal{D}^j$  represents that  $\hat{y}^j$  outperforms  $\tilde{y}^j$  when using  $j$  rationales. Finally, we introduce a penalizing factor  $P^j$  when  $j \neq \text{argmax}(\text{scores})$  to penalize gradients from less-performing cases. The objective for each component reacts to the two different cases differently during training, which we summarize as follows:

The final joint objective and expected cost is defined as:

$$\begin{aligned} \mathcal{L}_{gen} &= \sum_{j=1}^J \sum_{t=1}^T \mathcal{D}^j \left[ \mathcal{D}^j f(\hat{c}_t^j) + (1 - \mathcal{D}^j) f(\tilde{c}_t^j) \right] \\ f(c_t^j) &= -\log(s_t) \cdot c_t^j - \log(1 - s_t) \cdot (1 - c_t^j) \\ f_y(\hat{y}_i^j) &= -y_i \cdot \log(\hat{y}_i^j) \\ \mathcal{L}_{sel} &= \sum_{j=1}^J -\mathcal{D}^j * \text{scores}^j * \log(\text{scores}^j) \\ \mathcal{L}_{enc} &= \sum_{j=1}^J P^j \left[ \mathcal{D}^j f_y(\hat{y}^j) - (1 - \mathcal{D}^j) f_y(\tilde{y}^j) \right] \end{aligned} \quad (3.6)$$

$$\mathcal{L}(\mathbf{r}, \mathbf{x}, e_1, e_2, y) = \mathcal{L}_{gen} + \mathcal{L}_{sel} + \mathcal{L}_{enc} \quad (3.7)$$

$$\min_{\theta e, \theta g, \theta s} \sum_{(\mathbf{x}, e_1, e_2, y) \in N} \mathbb{E}_{\mathbf{r} \sim \text{sel}(\mathbf{c}), \mathbf{c} \sim \text{gen}(\mathbf{x})} \mathcal{L}(\mathbf{r}, \mathbf{x}, e_1, e_2, y) \quad (3.8)$$

where  $N$  is the training set and  $\theta e$ ,  $\theta g$ ,  $\theta s$  are parameters used in the Encoder, Generator and Selector respectively.

Notice that the Generator and Encoder handle losses from adversary success cases differently. The Generator exploits the adversary success cases as a potential learning resource, and uses  $\tilde{r}$  to train the RNN model. However, the Encoder is trying to make good predictions based only on rationales from the Generator, so it will treat good rationales from the adversary Generator as false cases and try to maximize adversary losses.

Finally, if ground-truth labels or human interventions for rationales  $r_{r_e}$  are provided, then we change  $\mathbf{r}$  in Equation (2) and  $\hat{c}$  in Equation (6) to  $r_{r_e}$ . This can be understood as a simple sequential

**Table 3.1** 10 Types of relations classes. Besides ‘Other’, all other relation types are bidirectional

Relation Classes	
Cause-Effect	Instrument-Agency
Component-Whole	Member-Collection
Content-Container	Message-Topic
Entity-Destination	Product-Producer
Entity-Origin	Other

discriminatory model where the Generator and the Selector is guided by  $r_{re}$ . However, while keeping the reminder of the model the same, introducing  $r_{re}$  does not guarantee the Selector and Encoder will use  $r_{re}$  while reasoning.

## 3.5 Experiments

### 3.5.1 Dataset

We performed our evaluation experiments on SemEval 2010 Task 8 data. A collection of sentences is provided, where each sentence is labeled with the target entities and the relation between the entities.

The dataset contains 10,717 sentences, and 9 types of relationship plus an ‘Other’ class. Relationship types (except for ‘Other’) are expressed bi-directionally, making 19 classes in total. Following SemEval 2010’s protocol, we used the macro-F1 metric in the experiment, excluding all cases of the ‘Other’ class.

Besides the original SemEval 2010 Task 8 data, we asked 3 human annotators to label rationales used in the test set. Annotators were asked to choose 0-3 rationales for each test sentence based on their intuition. Note that we did not annotated the training set.

### 3.5.2 Experimental Setup

Here we describe the detailed parameters used in the experiment. We initialized word vectors using the GoogleNews<sup>1</sup> vector that is learned by word2vec algorithm. Word vector dimensionality is set to 300. We naively set the the recurrent unit size in the Generator to 50, and the CNN filter size in the Encoder 50 for each  $\ell$ . We use at most 3 rationales and set the penalizing factor  $P^j$  to 0.5 based on our analysis on the training set.

We used the Adagrad optimizer in our experiments - the learning rate is initially set to 0.01, and reduces by 10% after every iteration. During training, we sampled 50 times from the Generator, and

<sup>1</sup><https://code.google.com/p/word2vec>

re-sampled 50 times when training the Selector and Encoder.

### 3.5.3 Results

We present the comparison between our model and other models in Table 4.2. Dependency tree models are neural network models that utilize a grammar parser to predict relation classes. Different models utilize the dependency tree differently. For instance in MVRNN [Soc12a] they used the whole dependency tree, while in SPTree [MB16] they experimented using both the whole tree and only nodes on the path that connected the target entities. We refer to other models that work without using a dependency tree as independent models.

We explain the details of each model in the following:

1. **SVM** [RH10]: The basic SVM model with several hand crafted features
2. **RNN - Recursive Neural Network and MVRNN** [Soc12a; Soc12a]: Similar to the one used in Chapter 2. The Recursive neural network utilizes the constituent tree to make a sentence representation. As the model is originally for sentence classification, there are no extra modifications for relation extraction. In other words, in these two research, relation extraction from sentence is considered as a type of sentence classification task.
3. **FCM - Factor-based Compositional Embedding Models** [Yu14b]: In FCM, the author utilize the dependency tree and the annotated edges in the dependency tree. Similar to [Soc12a], where each node is constructed by children nodes, in FCM each entity node is constructed by the nodes that points to it in the dependency tree. After entity nodes are computed, both nodes are then projected to a final vector space for classification. In other words, FCM model can be seen as a method to create entity representations, and relation is defined by the product of entity representations.
4. **Hybrid FCM** [Gor15]: Improving from [Yu14b], the hybrid FCM model utilizes other words in the sentence to model the entities better. In the original FCM, entity representations are computed by the child node of the entity. In hybrid FCM, the goal is to learn how to associate latent nodes that are not directly connected to the entities.
5. **DepNN: Dependency-Based Neural Network** [Liu15a]: In DepNN, they proposed to use the shortest path between the entities, and the subtree of each node on the shortest path. Each node on the shortest path will be augmented through recursive neural network by the node in subtree. Finally, the shortest path will be considered as a short sentence and use a convolutional neural network similar to [Kim14a] to make the prediction.
6. **DRNNs** [Xu15]: In DRNNs, the authors combines the shortest dependency path with negative sampling. The nodes and the grammar labels on the shortest dependency path is converted

to a short sequence, and then applied a convolutional neural network similar to [Kim14a] to obtain a sentence embedding. However, different from most approach, DRNN considers the shortest path from two directions, and compiles the result by taking the most significant class in the softmax functions from both direction. Note that in order to match the direction, the prediction in reversed  $e_2$  to  $e_1$  is reversed again to map with  $e_1$  to  $e_2$ .

7. **SPTree** [MB16]: The goal for SPTree is different from others, where the authors proposes a end-to-end entity and relation extraction model. That is, the model first extracts entities then predict the relationship between them. They begin the process by a recurrent neural network model to determine which words should be used as entities using the BILOU encoding proposed by [RR09], which is commonly used in Name-Entity-Recognition tasks. Each words will be given a BILOU label embedding, and the embedding will be associated to the word in a recursive neural network to predict the relationship. For SemEval2010 task 8 data used in this work, the recurrent neural network part is ignored and only the recursive neural network part is used as entities in this data is given.
8. **CNN** [Zen14]: In this CNN model, the author approach the relation extraction similar to how [Kim14a]. However, besides the word level features (word vectors) used in [Kim14a], this CNN model considers: Part-of-Speech tags (POS), word vectors of hypernym word in WordNet, and finally position to entities. Position to entities is converted to vector representation for all possible distance instead of scalar value.
9. **Stackforward, Vote-bidirect, Vote-backward** [NG15a]: In this model, the author combines a recurrent neural network model and a convolutional neural network model. In the **Stackforward** method, they use utilize a similar strategy in the second chapter of this thesis by using a CNN to first generate phrase vectors, and treat the phrase vectors as a sequence and combine them by a RNN. In **Vote-bidirect and backward**, the CNN and RNN model are detached, and the final prediction is obtained by taking the max argument from the sum of the softmax functions from both models. **Bidirect and backward** refers to how the order of how the RNN model process the sentence.
10. **ATT-Input-CNN, ATT-Pooling-CNN: Attention CNN** [Wan16]: In attention CNNs, the author experimented to include attention weights to relation extraction. Each non-entity words are mapped to the entities to obtain an attention score, and the final sentence embedding is obtained by a attention augmented CNN. By using attention, their model can also show the highlights of each sentence by plotting the attention scores for each word. Besides attention, instead of classification, this model is closer to clustering as the goal is to cluster sentence of same relation class to the relation in a vector space. Accordingly, this work is trained by negative sampling.

**Table 3.2** Comparisons with benchmarking models

<b>Classifier</b>	<b>F1</b>	<b>Classifier</b>	<b>F1</b>
<b><i>Manually Engineered Models</i></b>		<b><i>Independent Models</i></b>	
SVM[RH10]	82.2	CNN & softmax [Zen14]	82.7
<b><i>Dependency Tree Models</i></b>		Stackforward[NG15a]	83.4
RNN [Soc12a]	77.6	Vote-bidirect [NG15a]	84.1
MVRNN [Soc12a]	82.4	Vote-backward [NG15a]	84.1
FCM [Yu14b]	83.0	ATT-Input-CNN [Wan16]	87.5
Hybrid FCM [Gor15]	83.4	ATT-Pooling-CNN [Wan16]	88.0
DepNN [Liu15a]	83.6	<b><i>Rationale-based model</i></b>	<b>89.5</b>
DRNNs [Xu15]	85.6		
SPTree [MB16]	84.5		

**Table 3.3** Comparison with proxy rationale models.

<b>Rationale Models</b>	<b>Relation F1</b>
Proxy attention model	87.5
<b><i>Proxy rationales + Encoder</i></b>	
One rationale from proxy	84.5
Two rationales from proxy	85.3
Three rationales from proxy	87.8
Our rationale-based model	<b>89.5</b>

Besides the general goal of improving F1 score, we also evaluated the generated rationales qualities. Due to the size of the dataset, we were not able to annotate all samples manually - in order to handle this, we make an assumption that an Encoder will perform better if provided with rationales that are more interpretable, that is, rationales which are semantically closer to the entities and the desired goal. In effect, we utilize a set of identically-initialized Encoders, and evaluate rationales from different rationale finding approaches through a naive way by simply comparing the corresponding F1 scores from each Encoder.

However, since there are few comparable rationale models in this field, we created a proxy rationale model for comparison. We adapted a CNN model similar to [Kim14b], and added attention weights calculated by the similarity with  $(e_1, e_2)$ . To extract rationales from the proxy model, we first trained the CNN model, then extracted the top-weighted text fragments in the model as rationales. The proxy rationales were finally sent to the Encoder to compare with our model. We experimented this idea with 1, 2 and 3 rationales extracted from the proxy model, and results are shown in Table 3.3.

Finally, we summarized a few samples of rationales chosen from the test set in Table 4.5. From the table, we see that the selected rationales are similar to those that a human reader might use to

**Table 3.4** List of words/phrases that are selected as rationales by our model. Additional words are denoted as ... if they are located between entities and rationales but are not selected as rationales.

Relation Class	Selected rationales with target entities
Cause-Effect	congestion <sub>e1</sub> caused delays <sub>e2</sub> , bombing <sub>e1</sub> ... resulted in ... deaths <sub>e2</sub> devastation <sub>e1</sub> caused by the ... tsunami <sub>e2</sub> , anxiety <sub>e1</sub> caused by the accident <sub>e2</sub>
Component-Whole	carriages <sub>e1</sub> of ... train <sub>e2</sub> , title <sub>e1</sub> of the book <sub>e2</sub> , the kitchen <sub>e1</sub> has a fridge <sub>e2</sub> horn <sub>e1</sub> has ... end <sub>e2</sub> , this ... machine <sub>e1</sub> has ... reels <sub>e2</sub> , the shank <sub>e1</sub> of the hook <sub>e2</sub>
Content-Container	object <sub>e1</sub> was ... bag <sub>e2</sub> , wallet <sub>e1</sub> was inside ... locker <sub>e2</sub> , a mouse <sub>e1</sub> ... found in ... loaf <sub>e2</sub> bottle <sub>e1</sub> ... of ... dust <sub>e2</sub> , basket <sub>e1</sub> was full ... faces <sub>e2</sub> , the backpack <sub>e2</sub> contained a ... computer <sub>e2</sub>
Entity-Destination	yeast <sub>e1</sub> into ... tubes <sub>e2</sub> , niece <sub>e1</sub> moved into... apartment <sub>e2</sub> ... he put ... forkful <sub>e1</sub> ... mouth <sub>e2</sub> , hero <sub>e1</sub> enters into the building <sub>e2</sub>
Entity-Origin	artwork <sub>e1</sub> ... from ... evening <sub>e2</sub> , crisis <sub>e1</sub> originated in ... sector <sub>e2</sub> takara ... is a ... plum <sub>e1</sub> wine <sub>e2</sub> , the soul <sub>e1</sub> departs from the body <sub>e2</sub>
Instrument-Agency	marking birds <sub>e1</sub> ... researchers <sub>e2</sub> , this technique <sub>e1</sub> ... used ... websites <sub>e2</sub> technician <sub>e1</sub> uses ... emulator <sub>e2</sub> , shaman <sub>e1</sub> cured ... with herbs <sub>e2</sub> , man <sub>e1</sub> attacked ... with a ... bat <sub>e2</sub>
Member-Collection	chief <sub>e1</sub> of ... police <sub>e2</sub> , dean <sub>e1</sub> of the faculty <sub>e2</sub> , father mother <sub>e1</sub> was ... member ... team <sub>e2</sub> clowder <sub>e1</sub> of cats <sub>e2</sub> , an anthology <sub>e1</sub> of ... poems <sub>e2</sub> , the core ... body <sub>e1</sub> of ... traders <sub>e2</sub>
Message-Topic	book <sub>e1</sub> provides ... role <sub>e2</sub> , this figure <sub>e1</sub> illustrates ... results <sub>e2</sub> , the talk <sub>e1</sub> was about ... operations <sub>e2</sub> details <sub>e1</sub> about ... interview <sub>e2</sub> , the ... episode <sub>e1</sub> ... documented on video <sub>e2</sub>
Product-Producer	message <sub>e1</sub> from ... friend <sub>e2</sub> , paper <sub>e1</sub> co-authored by ... staff <sub>e2</sub> , blisters <sub>e1</sub> are caused by antibodies <sub>e2</sub> , cooper <sub>e1</sub> makes leak ... barrels <sub>e2</sub> , the researchers <sub>e1</sub> produced a report <sub>e2</sub>

infer the relationship between entities.

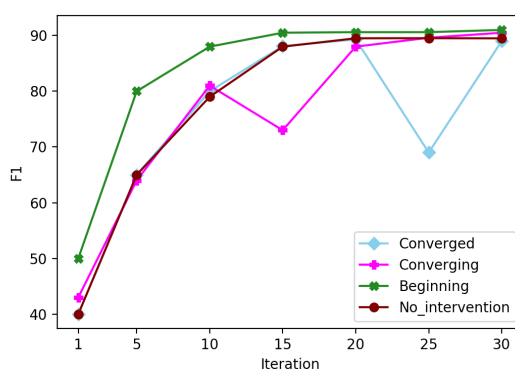
### 3.5.4 Results with Ground Truth for Rationales

Finally we evaluate our model when human annotators are included to provide ground-truth for rationales. While we were not able to perform detailed labeling, we naively generated ground-truth for rationales following the guidelines provided by SemEval 2010 task 8. We list the words that were used as ground-truth for rationales in Table 3.5. The Generator and Selector were trained in a supervised fashion if a ground-truth rationale was contained in a training sentence. Training samples that did not contain ground-truth rationales were trained in an unsupervised manner.

We injected ground-truth at three different training stages of the model: 1) converged, 2) con-

**Table 3.5** List of ground-truth rationales used

Relation Class	Selected Words
Cause-Effect	from, cause, caused, causes
Component-Whole	of
Content-Container	in
Entity-Destination	to, into
Entity-Origin	from
Instrument-Agency	use, uses, used, using
Member-Collection	of, is
Message-Topic	of, about
Product-Producer	make, makes, made



**Figure 3.5** F1 results using different intervention strategy.

verging, 3) from the beginning. Providing ground-truth to converged and converging models is intended to simulate models at production stage, and where human annotators are included only for final adjustment. Intervention performed at the beginning is designed to see how the model performs when it is closer to a fully supervised model.

We show the results in Figure 2. We can see that, when providing rationale labels at the beginning, the model learns much faster. Note that there is a performance drop in the converged and converging models immediately after ground-truth is provided at iteration 15 and 25 respectively. This result is expected since the input to the Encoder changes after introducing the labeled rationales, and this component requires a few iterations to assimilate the changes. Eventually, models with ground-truth labels achieve slightly better performance, typically a 1-1.5% improvement on F1 score.

Finally, we present a few example cases that were originally predicted incorrectly but that were subsequently corrected after manual labeling of rationales. Rationales are denoted in bold below:

**Example sentence:** Instrument-Agency( $e_2, e_1$ ):

**Before intervention** - The [generator] $e_1$  **creates** electricity using much the same [principle] $e_2$  as the

*alternator on your car ( depending on the turbine type )*

(predicted as Cause-Effect( $e_2, e_1$ ))

**After intervention** - *The [generator] $e_1$  creates electricity **using** much the same [principle] $e_2$  as the alternator on your car ( depending on the turbine type )*

**Example sentence:** Instrument-Agency( $e_2, e_1$ ):

**Before intervention** - *The [manufacturer] $e_1$  **assembles the** order using [parts] $e_2$  supplied by his preferred supplier , and ships the order to the retailer.*

(predicted as Product-Producer( $e_2, e_1$ ))

**After intervention** - *"The [manufacturer] $e_1$  assembles the order **using** [parts] $e_2$  supplied by his preferred supplier , and ships the order to the retailer."*

## 3.6 Discussion

### 3.6.1 Rationale Labeling

In the previous section we chose to label the rationales using a naive approach. The major drawback was that human language is complicated and often there exists several ways to express a relation class. While our naive approach only considers only very basic cases, future research would benefit from detailed annotated rationales for better generalizability.

### 3.6.2 End-to-End Training and Combining Components

At early stages of this research, we experimented with combining the Selector and Generator, and also tried end-to-end training. However we noticed that these often led to worse performance. We reason that this is because, in both setups, the selected rationales are heavily biased by the first few training epochs. However, this might not apply to cases where ground-truth for rationales is provided.

### 3.6.3 No-Rationale Cases

We notice that there are few cases where the model might provide ambiguous insight due to no good rationales being present, such as in the following example. The selected rationale, which is not useful, is shown in bold:

**Component-Whole( $e_2, e_1$ )** - *At the bottom of the [church] $e_1$  [steps] $e_2$  **were** three brown parishioners; two more were perched precariously on the railing of the deck.*

We were focussed only on selecting rationales in this work; however it may be possible to handle the no-rationale case in future work by introducing a mechanism that turns the Generator and

Selector off under certain circumstances.

### **3.6.4 Potential Uses**

Besides relation extraction, another potential use-case for our model is in generating rules for relation extraction. It should be possible to automatically discover sets of rules for relation extraction by supplying the Generator and Selector with carefully chosen test cases. This may be useful for knowledge base construction since building extraction rules manually becomes infeasible as the number of possible relation classes increases.

## **3.7 Conclusion**

In this paper, we proposed a novel generative adversarial approach for entity relation class prediction. In our model, we first generate a set of candidate rationales inside a generative RNN, and then predict the relationship between entities using a discriminative model that exploits the rationales. The generative process in our model can be turned into a semi-supervised model that is capable of incorporating human annotations. This not only boosts prediction accuracy but also provides a way for users to easily interpret and adjust the model. Quantitative analysis demonstrated that our model provides better performance than state-of-the-art dependency-tree and independent models. In addition, we showed that our model is able to select rationales that make sense to humans when combined with target entities.

# STABILIZING RATIONALE SELECTION

## 4.1 Introduction

In this chapter, we will reiterate the rationale-based relation model to illustrate the training challenges of the model.

The goal of sentence-level entity relation extraction is to infer how two target entities are semantically associated in a sentence. It is a core NLP function that supports many high level tasks such as information extraction and knowledge graph population [Hen09; Niu12]. A generative model will first extract the most representative fragments in the sentence that expresses the relation first, and augments the classifier with the rationales. These representative fragments are called rationales as per [Zha16b; Lei16; Hsu18], and can be used to justify the results.

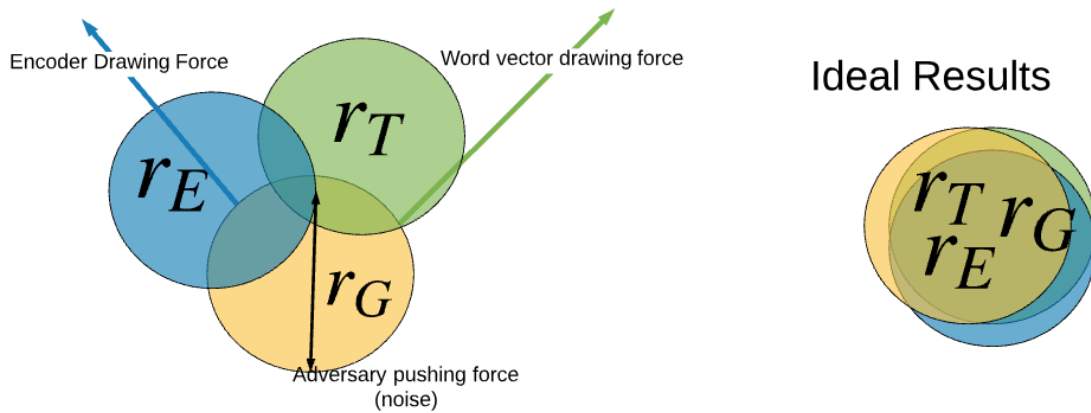
We illustrate an example of rationale-based models using an example sentence which expresses a Instrument-Agency( $e_2, e_1$ ) relationship between the given target entities  $e_1$  = “attacker” and  $e_2$  = “instrument”:

**Rationale based models:** “*She had struggled violently with her [attacker] $e_1$  , who **killed her with a blunt [instrument] $e_2$ .**”*

**(words marked in bold and the entities are the major sources to infer the relation)**

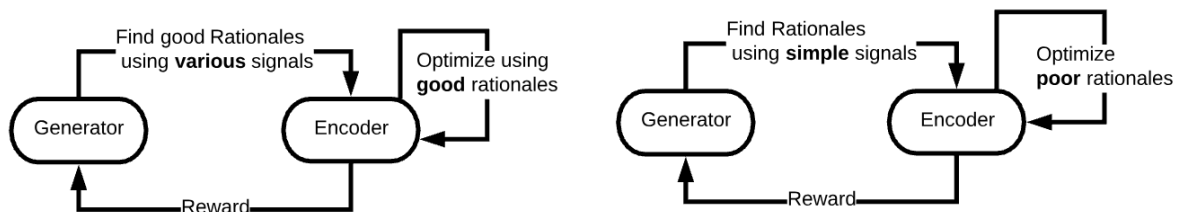
The fundamental structure of the rationale-based relation extraction model is a the Generator-

Encoder structure. The Generator extracts rationales from an input sentence in an unsupervised fashion; the Encoder, which is supervised, predicts the relation class utilizing the rationales. Since the Encoder is supervised hence more regulated, the key for the rationale-based models to success is hence mainly relying on the success of the Generator. There are two major factors to decide the outcome of the Generator: a good balance between the Encoder and introduced noise, and a drawing force towards the ground-truth rationales that human readers would use (Figure 4.1). The first factor decides whether the Generator and Encoder can converge to a nice coverage of rationales, and the second factor decides the interpretability and usefulness of the rationales.

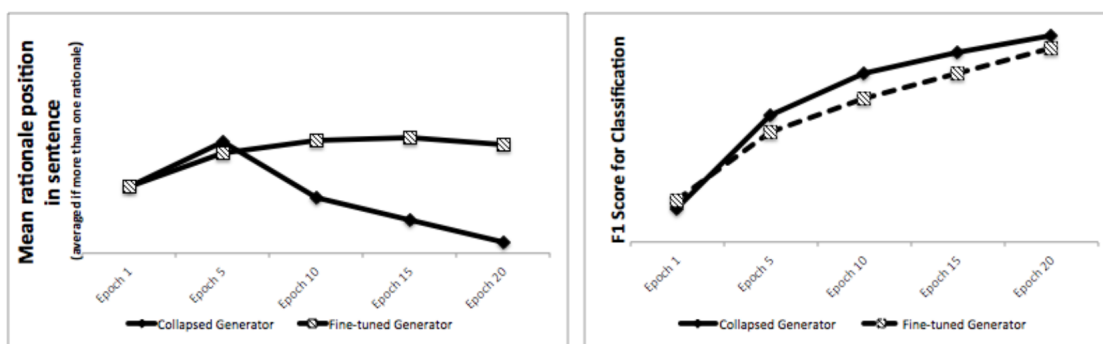


**Figure 4.1 Left Figure:** We illustrate the relation of rationales of different perspectives in a simple way: rationales the Encoder favors  $r_E$ , rationals generated by Generator  $r_G$  and finally ground-truth rationales  $r_T$ . There are multiple drawing forces that draws the Generator. **Right Figure:** The ideal results after training

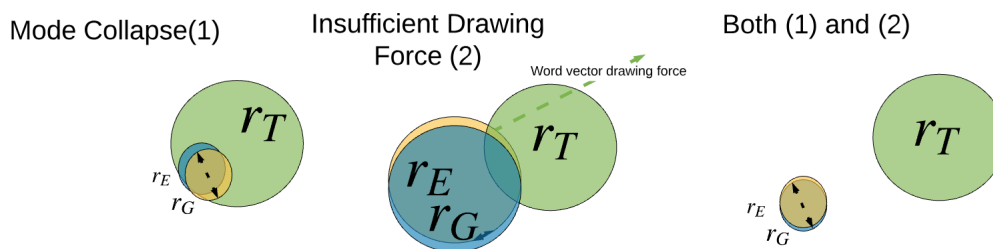
Configuring the right balance between the Encoder and noise is difficult and is affected by the randomness introduced by the alternative training process required in this structure. This is a commonly known issue called mode collapse in this line of research [Che16; Che17; DH18]. Mode collapse is a paradox where the Generator failed to extract diversified rationales because the optimization function does not encourage so(Figure 4.2). A collapsed Generator is not capable of providing meaningful rationales since it extracts the same rationales repeatedly regardless of the context. However, a point worthy to point out as shown in Figure 4.3, a collapsed Generator does not always lead to poor classification performance in the Encoder when compared to a finely converged Generator.



**Figure 4.2 Left Figure:** The theoretical outcome of a successful result of our rationale-based relation extraction model - the Generator outputs diversified but convincing outputs to the Encoder. **Right Figure:** The outcome that happens frequently during our training process where the Generator decision factors collapsed to a single criteria(mode)



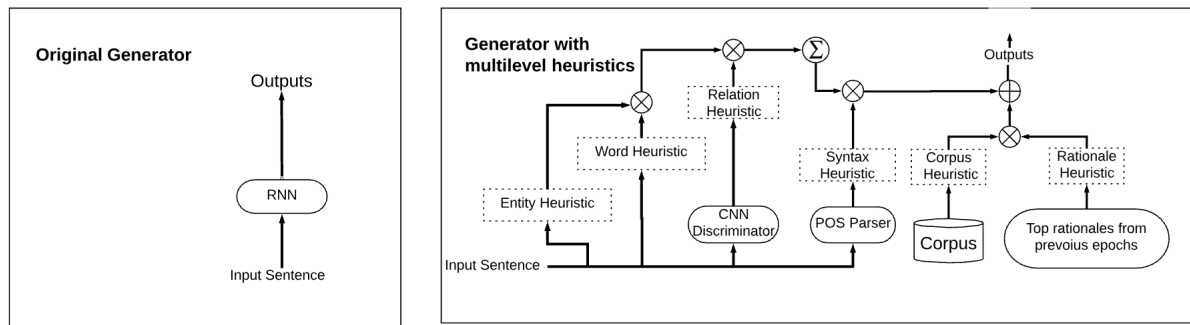
**Figure 4.3** A Generator can converge at collapsed rationales but still lead to good classification results. **Left Figure** shows the rationale distribution of a collapsed and a well-converged Generator that uses a RNN. The well-converged Generator evenly selects rationales from a sentence, while the collapsed Generator selects rationales close to the beginning of the sentence. This is because the best scenario for Encoder is that some rationales are repetitively selected and optimized in every training iteration, which is difficult for the Generator. Eventually, the Generator learns to exploit position features since it is less diversified than words. **Right Figure** shows that a collapsed Generator does not necessarily deteriorate the classification performance.



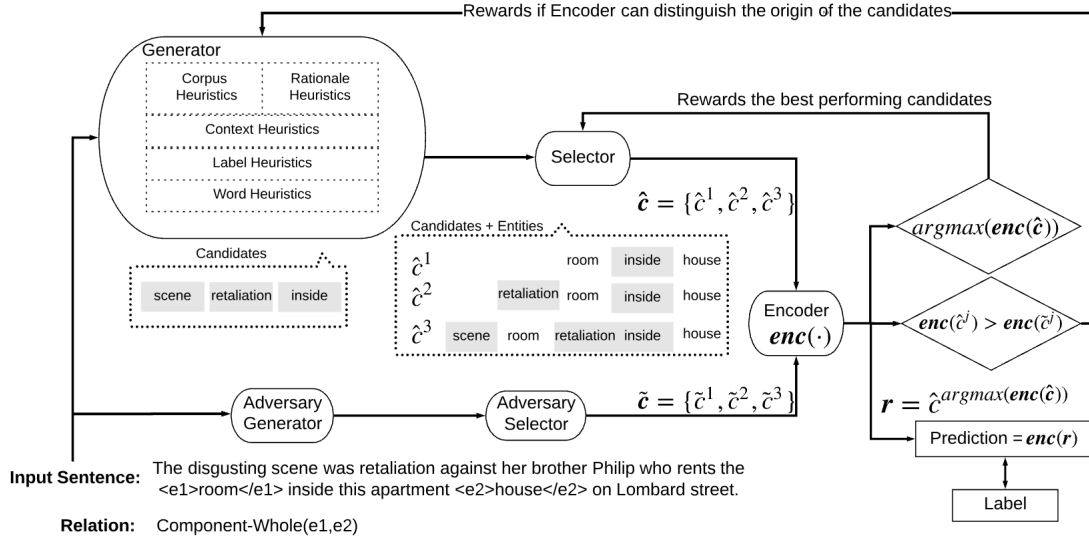
**Figure 4.4 Left Figure:** The Issue of unbalanced Encoder and noise. The variety of rationales shrinks as the Generator collapsed to utilize only monotonous signals to decide rationales. **Middle Figure:** The issue of poor drawing force to good rationales, where the Generator converged to generate a poor collection of rationales. **Right Figure:** Results of insufficiently tuned model where both mode collapse and the poor drawing force happened.

While mode-collapse affects the balance of the model and posts a challenge to the first success factor of the Generator, the lack of ground-truth rationales creates another challenge as well. In previous chapter, we made the assumption that the pre-trained word vectors and relation labels will have enough guiding power to lead rationales converge toward the right direction. However, these implicit information are not always sufficient to consistently guide the model to select useful rationales in our experiments. During training, the model often failed to utilize rationales that a human reader would use and hence require careful tuning of the model.

In this chapter, we aim to improve the rationale-based models for relation classification by introducing semi-supervised capacity to strengthen the drawing force, and additional regularizing force to the Generator to prevent mode collapsing. In this updated Generator, we develop a multilevel heuristic to improve the context-focused RNN from the original framework. The updated Generator jointly considers features from different levels of the dataset such as contextual words, target entities, sentence syntax features, rationale-relation closeness, and global word distribution in corpus. We showed the final rationale-based relation extraction model with the updated multilevel heuristic mechanism in Figure 4.5 and Figure 4.6.



**Figure 4.5** Figure on the left shows the original Generator in previous chapter, and the right shows the improved multilevel heuristics Generator. The Word heuristic measures the relative closeness of each word in the sentence to every relation class. The Entity heuristic uses the target entities to regulate the Word heuristic. The Relation heuristic contains predictions of the relation from a standalone convolutional neural net model and is also used used to regulate the Word heuristic. The Word heuristic is then summed and weighted by the Syntax heuristic, which considers both, the word distance to entities, and Part-of-Speech labels. The Corpus heuristic is obtained by word-relation distribution in the corpus. The Rationale heuristic is the semi-supervised component which is based on rationales from the previous training iteration.

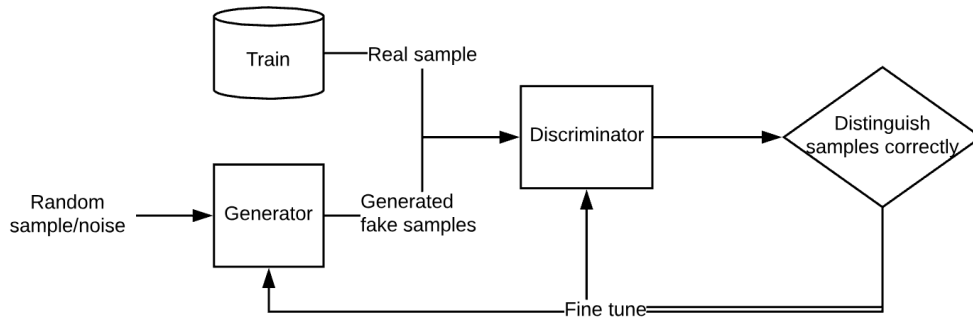


**Figure 4.6** The major improvement lies in the Generator step by changing the Generator from a RNN to several simple feed-forward Neural Networks that optimizes against different level of text features. By specifically separating the optimization goal, we hypothesize it will alleviate mode collapse issue happened while using recurrent neural network where the Generator is solely guided by the Discriminator.

## 4.2 Related Work

Research for improving generative neural networks have gained much interest in the research community due to the effectiveness and versatility of these models. The Generative Adversarial Network (GAN) [Goo14] is one of the most representative model in this field. [Goo14] proposed a modified generative model using an adversarial minimax-game formulation (Figure 4.7). The objective of GAN is to jointly train a Generator and Discriminator, where the loss for Discriminator and the reward for Generator comes from failed attempts to determine ground-truths from the emulated results created by Generator. Several research [Ree16; Zhu17; Yi17; Tul17; MO14] had shown the adversarial process in GAN is effective at generating fake data that are indistinguishable to the original train data (Figure 4.8).

However, converging the adversary process in GAN is a major challenge in this line of research. More specifically, there exists a mode collapse issue in the process. Mode collapse is the situation where the Generator generates the same output regardless of the input. The reason for mode-collapse to happen is due to the training order in the adversary process and the loss function it used. Under extreme cases, the Generator will learn to reach the best performing output first, which will be the same point for all inputs given that the Discriminator does not change. After Generator is updated, the Discriminator will then attempt to penalizes the Generator for the incorrect outputs. However, since the results from the Generator are all identical, there's not much different directions for the gradients and hence it will fail to push the Generator toward any directions [Sal16; Met16]



**Figure 4.7** The basic generative adversarial network framework [Goo14]. The Generator will try to make fake samples based on some raw inputs, and the goal for Discriminator is to distinguish the fake samples from true samples. Take a text to image example where the text is "A yellow bird on a tree", the Generator will map the text to a generated image, then feed the generated image with the real image that has a yellow bird on a tree to the Discriminator. The Discriminator is trained using the true labels, and the Generator is rewarded/penalized based on successfully deceiving the Discriminator or not.

(Figure 4.9).

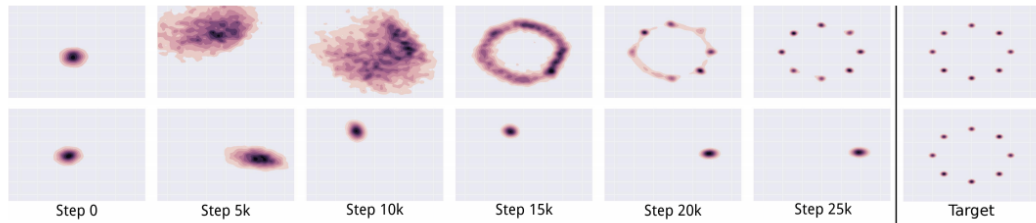
Research to avoid mode collapse can be roughly divided into two categories: architecture-oriented and divergence-oriented. Architecture-oriented models focused on changing how the Generator and Discriminator are connected or add extra information to restrain the model, and divergence-oriented focused on optimizing the loss function to ensure diversity.

We begin with architecture-oriented models. For example, InfoGan [Che16] proposed to minimize the mutual information of auxiliary loss in GAN to produce disentangled unsupervised representations. [Sal16] proposed to use semi-supervised training and label-smoothing to help GAN converge to Nash Equilibrium between the Generator and the Discriminator. [DH18] proposed RF-GAN to implicitly regularize the discriminator using representative features extracted from the data distribution. In VEEGAN [Sri17] 4.10, the authors suggested to jointly train an extra Reconstructor with the Generator which is an approximate reverse action of the Generator to encode noise.

Another group of methods have aimed to reduce mode collapse in generative models by modifying the joint loss functions used between the Generator and Discriminator. In [Arj17], the authors theoretically showed that the Kullback-Leibler (KL) divergence often used in GAN was one of the source of instability. We show part of the reasoning in the following where  $P$  and  $Q$  are two distributions that overlaps when  $\theta = 0$ .



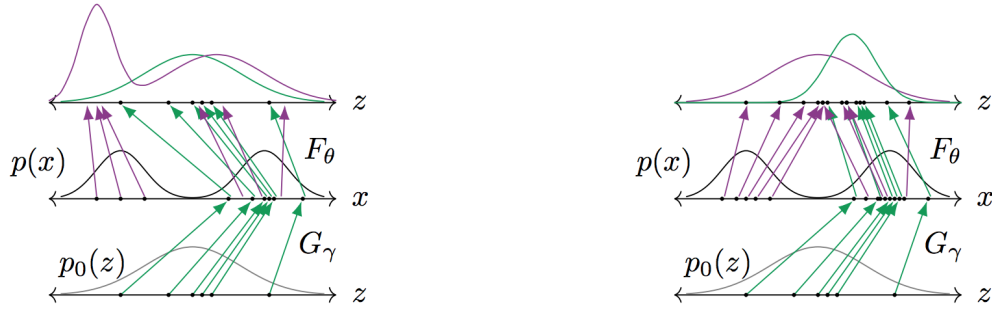
**Figure 4.8** Example output from [Ree16]. The text is the input and the outputs are generated image that matches to the sentence meaning.



**Figure 4.9** Mode collapse visualization from [Met16]. The data is a collection of 2D Gaussian Distributions, the final column is the ground-truth ( a dotted circle). The upper row shows the more expected results, where a well-trained GAN model will be able to produce results similar to the ground-truth. The bottom row represents a standard collapsed GAN where the Generator produces one type of data pattern (only points of a certain region in the ground-truth) in a period of training epochs.

When P and Q does not overlap:

$$\begin{aligned}
 & \forall (x, y) \in P, X = 0 \text{ and } y \sim U(0, 1) & \forall (x, y) \in Q, 0 \leq \theta \leq 1, \text{ and } y \sim U(0, 1) \\
 D_{KL}(P||Q) &= \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty & D_{KL}(Q||P) &= \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{0}{1} = -\infty \\
 D_{JS}(P, Q) &= \frac{1}{2} \left( \sum_{x=0, y \sim U(0,1)} \cdot \log \frac{1}{2} + \sum_{x=0, y \sim U(0,1)} \cdot \log \frac{1}{2} \right) = \log 2 \\
 W(P, Q) &= |\theta|
 \end{aligned} \tag{4.1}$$



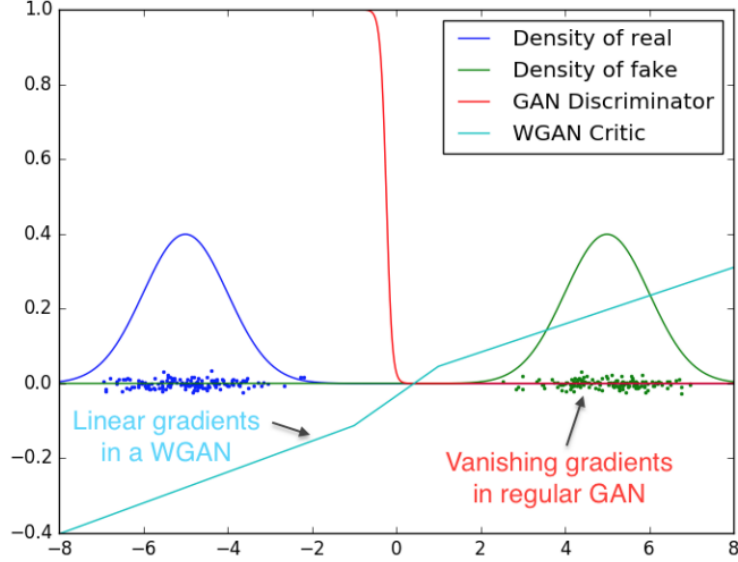
**Figure 4.10** VEEGAN model[Sri17].  $F_\theta$  represents the Reconstructor. Figure on the left shows when mode collapsed is happening, inverting the results of the Generator through Reconstructor  $F_\theta$  should show a different distribution to the input data. Figure on the right shows Reconstructor can also help tune the Generator by serving as an autoencoder.

When P and Q overlap:

$$\begin{aligned}
 \forall(x, y) \in P, X = 0 \text{ and } y \sim U(0, 1) & \quad \forall(x, y) \in Q, X = 1 \text{ and } y \sim U(0, 1) \\
 D_{KL}(P||Q) = 0 & \quad D_{KL}(Q||P) = 0 \\
 D_{JS}(P, Q) = 0 & \\
 W(P, Q) = |\theta| & \tag{4.2}
 \end{aligned}$$

The above discovery shows KL-divergence and JS-divergence both encountered non-continuous loss curve causing the loss not differentiable. Also, KL-divergence is not symmetric which leads to GAN optimize only the Generator or the Discriminator better but not both. Based on this discovery, [AB17] proposed Wasserstein distance to measure the similarity between the fake and real data to make stabler Generator. [Gul17] introduces the gradient penalty for regularizing the divergence as an alternative to gradient clipping used in [AB17] to capture stronger movements in gradients. In DRAGAN [Kod17], they proposed to regulate sharp gradients in the Discriminator, which often happens at some undesired *local equilibria* between Generator and Discriminator. [Rot17] proposed another type of regularization that breaks down the divergence into different Discriminator output to achieve stability.

We position our work closer to the first category since we can view the multilevel heuristics used in our model close to a combination of models in the first category. The Word and Corpus heuristic are derived from data distribution similar to RFGAN[DH18], and the Relation and Rationale heuristic are substantially close to the semi-supervised learning techniques used in [Sal16]. Furthermore, the frequency based Corpus and Rationale heuristics can provide a stronger navigation than word vectors toward ground-truth rationales. However, our work is not exclusive to the second category



**Figure 4.11** Wgan results[Arj17]. The figure shows in standard GAN the gradients at Discriminator side could vanish hence stop training.

as the Word heuristic included idea close to [Rot17] where the divergence in the Word heuristic maps to label level in Discriminator instead of treating the Discriminator as a whole.

### 4.3 Proposed Model

In this section, we describe each component of the framework. We begin by reiterating the Encoder to formalize the goal of relation classification.

#### 4.3.1 Encoder

Given an input training tuple  $(x, e_1, e_2, y)$ , where  $x = \{x_t\}_{t=1}^T$ , and  $y$  is the one-hot  $L$  dimensional vector representing the relation class. The goal of the Encoder  $enc(r, x, e_1, e_2)$  is to produce a probability distribution  $\hat{y}$  that approximates to  $y$  as formulated as follows ( $N$  is the size of the training set):

$$loss = \sum_{i=1}^{|N|} -y_i \cdot \log(\hat{y}_i) \quad (4.3)$$

The Encoder produces  $\hat{y}$  by encoding  $(r, x, e_1, e_2)$  as a vector representation and then projects it to the  $y$  space. This vector representation can further break down to a sentence vector  $V_x$  and a rationale vector  $V_r$ . We produce  $V_x$  through CNN using the same model as [Kim14b] which encodes

the sentence through a continuous window approach, where  $K$  are the set of sizes for the windows. To produce  $V_r$ , we implement a RNN model using GRU cells that reads the rationales and target entities. Since RNN is order sensitive, the rationales and the entities are sorted based on their position in the sentence before applying the RNN model.

$$\begin{aligned}
 V_x &= CNN(\mathbf{x}) & \hat{y} &= enc(\mathbf{r}, \mathbf{x}, e_1, e_2) \\
 V_r &= RNN([\mathbf{r}, e_1, e_2]) & &= softmax(W_{enc} \cdot [V_r, V_x, e_1, e_2] + b_{enc})
 \end{aligned}
 \tag{4.4}$$

### 4.3.2 Generator

The goal of the Generator  $gen(x)$  is to derive a set of binary variables  $\mathbf{c} = \{c_t\}_{t=1}^T$  by sampling from probability scores  $\mathbf{s} = \{s_t\}_{s=1}^T$ , where  $c_t \in [0, 1]$  indicates whether  $x_t$  is chosen as a candidate rationale and  $s_t$  is computed by the multilevel heuristics. We represent it as  $\mathbf{c} \sim gen(x)$ .

In contrast to our previous Generator which uses an RNN model for Generator, we adopted a set of logistic regression models with several feed-forward networks to facilitate the Generator. The reason being, an RNN model can be difficult to train and might be susceptible to mode collapse if not carefully tuned. In the Generator, the number of logistic regression models is equal to  $L$  which is the number of classes in  $y$ . Each word in the sentence is first moderated by the Entity heuristic  $he$ , which is obtained from the target entities and fed through the logistic regression models to obtain the Word heuristic  $hw = \{\{hw_t^l\}_{l=1}^L\}_{t=1}^T$ . Each  $hw_t^l$  can be considered as a probability of  $x_t$  being chosen as a rationale if the given sentence is of relation class  $l$ .

The second step is to compute the Relation heuristic  $hr = \{hr^l\}_{l=1}^L$  and the Syntax heuristic  $hs_t$  that are applied to the Word heuristic. The Relation heuristic initializes the relation with a random guess using the whole sentence. The heuristic is then trained through a CNN model which similar to the CNN model we used to compute  $V_x$ . The Syntax heuristic  $hs_t$  is obtained from a feed forward neural network that takes as input (1) word distance to target entities, and (2) Part of Speech (POS) tags. The output,  $s_t^{local}$ , is obtained by combining the Word, Entity, Syntax and Relation heuristics, and it represents local sentence information score.

The final score  $s_t$  is then computed by applying the Corpus and Rationale heuristics to  $s_t^{local}$ . The Corpus heuristic  $hc = \{hc^l\}_{l=1}^L$  contains binary values computed by sorting and selecting the frequent  $K$  words in each relation class in the training set. The Rationale heuristic  $hra = \{hra^l\}_{l=1}^L$  also contains binary values which are based on taking the top portion of the Word heuristic for each relation class, but the Word heuristic considered here is not moderated by entities when computing the Rationale heuristic. The Corpus and Rationale heuristics capture the association between the words and relations are in the dataset. To leverage this information, we construct a global heuristic,  $s_t^{global}$ , by computing an element-wise multiplication of  $hc$  and  $hra$ . Note that  $s_t^{global}$  can be

seen as a self-learning resource for semi-supervised learning because the Rationale heuristic is based on the results from previous iterations. Accordingly,  $s_t^{global}$  will not be used in testing and will only be included after several training iterations have completed. Finally, before applying  $s_t^{global}$  to the final score  $s_t$  we apply a dynamically computed weight factor  $CR$  to  $s_t^{local}$  and  $s_t^{global}$ . We formulate all the heuristics in the following equation,

$$\begin{aligned}
he &= \text{sigmoid}(W_{he} \cdot [e1, e2] + b_{he}) & s_t^{local} &= hw_t \cdot \mathcal{T}(hr) \cdot hs \\
hw_t &= \text{sigmoid}(W_{hw} \cdot (x_t \odot he) + b_{hw}) & & (\mathcal{T} \text{ stands for transpose function}) \\
hr &= \text{CNN}(x) & s_t^{global} &= hc \odot hra \\
hs_t &= \text{sigmoid}(W_{hs} \cdot [POS_t, position_t]) & s_t &= s_t \cdot (1 - CR) + s_t^{global} CR \\
CR &= \text{sigomid}(W_{CR} \cdot hr' + b_{CR}) & &
\end{aligned} \tag{4.5}$$

To summarize, the updated Generator jointly considers several aspects such as contextual words, target entities, word distribution in corpus, sentence grammatical features, and selected rationale-relation closeness to regularize rationale selection.

In the most simplistic Generator, the probability that  $x_t$  is chosen is conditionally independent of all other  $x$ . In other words, we can sample candidate  $c$  from a uniform distribution using the probability scores,  $s$ . However, we observe that the target rationales in the dataset often consist of few words - for example “room *inside* apartment”. Therefore, we added the following Equation (4) to limit the number of rationales selected,  $c$ , to be at most  $J$ .

$$\begin{aligned}
c_t &= \begin{cases} 1, \text{sort}(\{(s_t > \text{rand}(0, 1)) \cdot s_t\}_{t=1}^T)[1:J] \\ 0, \text{otherwise} \end{cases} & (4.6) \\
c &= \{c_t\}_{t=1}^T
\end{aligned}$$

### 4.3.3 Selector

The last component of the model is the Selector  $sel(c)$ , where the goal of the Selector is to decide the number of rationales best suited for the prediction. The Selector is facilitated by a simple feed-forward network that outputs a number from  $1 \sim J$  by scoring rationale sets of length ranging from  $1 \sim J$  based on the top ranked  $s_t * c_t$ . During training, all rationale sets will be forwarded to the Encoder, and the training label for Selector will be the size of the set that obtains the least training loss in Encoder. This Selector is identical to the one in [Hsu18] and is defined as follows:

$$\begin{aligned}
\hat{c}_t^j &= \begin{cases} 1, \text{sort}(\{c_t \cdot s_t\}_{t=1}^T)[1:j], j \in [1:J] \\ 0, \text{otherwise} \end{cases} \\
\hat{c}^j &= \{\hat{c}_t^j\}_{t=1}^T, \text{ where } \hat{c}_t^j = 1 \\
\text{score}(\hat{c}^j) &= W_c \cdot \text{enc}(\hat{c}^j, x, e_1, e_2) \\
\text{scores}^j &= \text{softmax}(\{\text{score}(\hat{c}^j)\}_{j=1}^J) \\
\text{scores}^j &= \begin{cases} 1, j = \text{argmax}(\text{scores}) \\ 0, \text{otherwise} \end{cases} \\
r &= \{x_t\}, \text{ where } \hat{c}_t^j = 1, j = \text{argmax}(\text{scores})
\end{aligned} \tag{4.7}$$

#### 4.3.3.1 Joint Objective and Adversarial Training

We formulate the joint loss function in the following to bind the components.

As described earlier, the goal of the Generator and the Selector is not to emulate  $x$  with  $r$  due to the nature of this research. In effect, the Generator and the Selector are not competing against the Encoder, but instead with an adversary Generator, for which we use a randomized approach. The adversary Generator will sample  $\tilde{c}$  randomly from  $x$ . The adversary rationales  $\tilde{r}$  will then be selected using  $\tilde{c}$  as per Equation (4).  $\tilde{r}$  will be passed to the Encoder to generate a projection  $\tilde{y}$  onto the target dimension. We compare  $\tilde{y}$  with  $\hat{y}$  in terms of their similarity to  $y$  and is denoted by  $\mathcal{D} \in \{0, 1\}$ .  $\mathcal{D} = 1$  when  $\hat{y}$  is closer to  $y$  compared to  $\tilde{y}$ , and is noted as a ‘model success case’, meanwhile  $1 - \mathcal{D} = 1$  for the opposite situation and is noted as an ‘adversary success case’. The objective for the Generator and Encoder reacts differently to the two different cases, where the Encoder optimize whichever rationales that performs better, and the Generator rewards the selected rationales in model success case and penalize in the other case. We further split  $\hat{y}$ ,  $\tilde{y}$  and  $\mathcal{D}$  into  $\hat{y}^j$ ,  $\tilde{y}^j$ ,  $\mathcal{D}^j$  and  $j \in [1:J]$ , where  $\mathcal{D}^j$  represents that  $\hat{y}^j$  outperforms  $\tilde{y}^j$  when using  $j$  rationales. Finally, we introduce a penalizing factor  $P^j$  when  $j \neq \text{argmax}(\text{scores})$  to penalize the gradients from poor performing cases. We summarize as follows:

$$\begin{aligned}
f(s_t^{j^l}) &= -\log(s_t^{j^l}) \cdot c_t^j \cdot y_i + \\
&\quad -\log(1 - s_t^{j^l}) \cdot c_t^j \cdot (1 - y_i) \\
f(c_t^j) &= -\log(s_t) \cdot c_t^j - \log(1 - s_t) \cdot (1 - c_t^j) \\
f_y(\hat{y}_i^j) &= -y_i \cdot \log(\hat{y}_i^j) \\
\mathcal{L}_{enc} &= \sum_{j=1}^J P^j \left[ \mathcal{D}^j f_y(\hat{y}^j) + (1 - \mathcal{D}^j) f_y(\tilde{y}^j) \right] \\
\mathcal{L}_{sel} &= \sum_{j=1}^J -\mathcal{D}^j * \text{scores}^j * \log(\text{scores}^j) \\
\mathcal{L}_{gen} &= \sum_{j=1}^J \sum_{t=1}^T \left[ \mathcal{D}^j f(\hat{c}_t^j) - (1 - \mathcal{D}^j) f(\tilde{c}_t^j) \right] + \\
&\quad \sum_{j=1}^J \sum_{t=1}^T \sum_{l=1}^L \left[ \mathcal{D}^j f(\hat{s}_t^{j^l}) - (1 - \mathcal{D}^j) f(\tilde{s}_t^{j^l}) \right]
\end{aligned} \tag{4.8}$$

The final joint objective and the expected cost is defined as:

$$\mathcal{L}(r, x, e_1, e_2, y) = \mathcal{L}_{gen} + \mathcal{L}_{sel} + \mathcal{L}_{enc}$$

**Table 4.1** 10 Types of relations classes. Besides ‘Other’, all other relation types are bidirectional

Relation Classes	
Cause-Effect	Instrument-Agency
Component-Whole	Member-Collection
Content-Container	Message-Topic
Entity-Destination	Product-Producer
Entity-Origin	Other

$$\min_{\theta_e, \theta_g, \theta_s} \sum_{(x, e_1, e_2, y) \in N} \mathbb{E}_{r \sim \text{sel}(c), c \sim \text{gen}(x)} \mathcal{L}(r, x, e_1, e_2, y) \quad (4.9)$$

where  $\theta_e$ ,  $\theta_g$ ,  $\theta_s$  are the parameters used in the Encoder, Generator and Selector respectively.

## 4.4 Experiments

### 4.4.1 Dataset

We evaluate our approach by performing experiments on the SemEval 2010 Task 8 dataset. A collection of sentences is provided with marked target entities and ground-truth for the relation between the entities.

The dataset contains 10,717 sentences, where 8000 entries are used in training. There are 9 types of relationships plus an ‘Other’ class (Table 4.1). Relationship types (except for ‘Other’) are expressed bi-directionally, making 19 classes in total. Following SemEval 2010’s protocol, we used the macro-F1 metric in the experiment, excluding all cases of the ‘Other’ class.

Besides the original SemEval 2010 Task 8 data, we asked 3 human annotators to label rationales used in the test set. Annotators were asked to choose 0-3 rationales for each test sentence based on their intuition. Each annotator is asked to roughly annotate one-third of the test set. Distribution of rationales are as follows: no-rationales:2%, 1-rationale:57%, 2-rationales:38% and 3-rationales:2%. Note that we did not annotated the training set.

### 4.4.2 Experimental Setup

We now describe in detail the parameters used in our experiments. We initialize the word vectors with the GoogleNews<sup>1</sup> corpus. The CNN filter size was initialized to 50. We use at most 3 rationales and set the penalizing factor  $P_j$  to 0.1 based on our analysis of the training set. Due to the lack of sufficient samples in certain relation classes, we ignored the relationship directionality while

<sup>1</sup><https://code.google.com/p/word2vec>

computing the Relation heuristic. Finally, the dimensionality of the POS tag vectors and the position vectors is set to 25, and the Generator is set to ignore nouns, articles and adjectives.

Finally, after training for 10 iterations, the Generator includes the output from the Corpus and Rationale heuristics. The  $K$  frequently used words in the Corpus heuristic are computed for each relation class, where the threshold for  $K_l$  is 10% of number of sentences of the given  $l$  relation class. The Rationale heuristic selects the top 25% of words,  $hw^l$ , output by the Word heuristic.

We used the Adagrad optimizer in our experiments and the learning rate is initialized to 0.01, and reduces by 10% after every iteration. During each training iteration, we sample 25 times from the Generator, and re-sample 25 times when training the Selector and the Encoder.

In addition, we ran confidence interval on both relation and rationale F1 scores. For rationale F1, similar to the experiments in Chapter 2, we used bootstrap methods with replacement to compute rationale F1 scores for the proxy models, the original rationale-based model and the updated rationale-based model. Similar to Chapter 2, we sampled 1,000 times for each model. For relation F1, we trained 1,000 copies of our model to generate confidence interval. The reason being that labels in this dataset is very unbalanced and hence the sampling process could cause major impact on the major F1 score. Note that to finish the experiments in a reasonable time we modified the algorithm to train faster (by using smaller training epochs and fewer samples), but it still takes more than a month to complete.

### 4.4.3 Results

We present a comparison of relation classification between our model and the state-of-the-art models in Table 4.2. Dependency tree models are neural network models that work with word dependency tree parsed by a grammar parser. Each model utilizes the dependency tree differently. For instance MVRNN [Soc12a] uses the whole dependency tree, while SPTree [MB16] experiments using the entire tree and the nodes on the path that connect the target entities. We refer to the models that do not use a dependency tree as independent models. More information of the benchmark models could be found in Chapter 3.

Table 4.3, presents a detailed comparison between the different variations of our model and other rationale-based models. We evaluate the rationale quality using F1 score, where precision and recall are computed by taking the intersection between the generated and the manually labeled rationales. Table 4.4 shows the detailed F1 score for each relation types.

Finally, we summarize a few samples of rationales chosen from the test set in Table 4.5. We observe that the extracted rationales are similar to the rationales chosen manually and they can be used to infer the relationship between entities.

**Table 4.2** Comparisons with benchmarking models

Classifier	F1	Classifier	F1
<b><i>Manually Engineered Models</i></b>		<b><i>Independent Models</i></b>	
SVM[RH10]	82.2	CNN & softmax [Zen14]	82.7
<b><i>Dependency Tree Models</i></b>		Stackforward[NG15a]	83.4
RNN [Soc12a]	77.6	Vote-bidirect [NG15a]	84.1
MVRNN [Soc12a]	82.4	Vote-backward [NG15a]	84.1
FCM [Yu14b]	83.0	ATT-Input-CNN [Wan16]	87.5
Hybrid FCM [Gor15]	83.4	ATT-Pooling-CNN [Wan16]	88.0
DepNN [Liu15a]	83.6	<b><i>Rationale Models</i></b>	
DRNNs [Xu15]	85.6	Proxy-Rationale-1[Hsu18]	84.5
SPTree [MB16]	84.5	Proxy-Rationale-2[Hsu18]	85.3
		Proxy-Rationale-3[Hsu18]	87.8
		Original Generator[Hsu18]	89.5
		Updated Generator	<b>90.7</b>

**Table 4.3** Comparisons between variations and with rationale models. The numbers in the parentheses are confidence interval. Note that we trained a faster version of our model by reduced sampling for confidence intervals to make the model trained within a reasonable time. To be specific about the modified settings, we changed epoch from 25 to 15, and sampling times from 25 to 10.

Rationale Models	Relation F1	Rationale F1
Proxy-Rationale-1	84.5	6.3 (6.59, 5.3)
Proxy-Rationale-2	85.3	22.2 (22.8, 21.5)
Proxy-Rationale-3	87.8	26.1 (26.9, 25.2)
Original Generator	89.5	33.2 (33.6, 32.1)
Updated Generator prior global heuristics $s^{global}$	89.5	37.1 (37.8, 36.3)
Updated Generator after global heuristics $s^{global}$	<b>90.7</b> (90.6, 90.0)*	<b>53.2</b> (54.2, 52.6)

## 4.5 Discussion and Future Goals

### 4.5.1 Logistic Regressions and RNN

The major difference between this work and the previous rationale-based research is that we designed the Generator without a sequential RNN model. Specifically, we developed a layer by layer process that considers different levels of text information for three reasons. First, in relation classification, the excess sequential modeling power from RNN is not required since rationales are often an extremely small span of words. Also, rationales are often strongly-related to POS tags which can be used to replace the sequence information. Second, the purpose of rationales in relation classification is to augment features and illuminate how the model derives the answer. This is different from other tasks like aspect-level sentiment classification in [Lei16] where the rationales are treated as

**Table 4.4** Detailed F1 score for each relations.

<b>Relation Type</b>	<b>F1</b>	<b>Relation Type</b>	<b>F1</b>
Cause-Effect(e1,e2)	94.9	Cause-Effect(e2,e1)	93.2
Component-Whole(e1,e2)	89.8	Component-Whole(e2,e1)	83.3
Content-Container(e1,e2)	89.7	Content-Container(e2,e1)	80.6
Entity-Destination(e1,e2)	94.5	Entity-Destination(e2,e1)	-
Entity-Origin(e1,e2)	89.9	Entity-Origin(e2,e1)	92.7
Instrument-Agency(e1,e2)	84.2	Instrument-Agency(e2,e1)	85.7
Member-Collection(e1,e2)	95.5	Member-Collection(e2,e1)	85.1
Message-Topic(e1,e2)	94.6	Message-Topic(e2,e1)	92.3
Product-Producer(e1,e2)	87.6	Product-Producer(e2,e1)	87.6

a compression of the input. Although more empirical experiments are required, we hypothesize that this simpler multilevel heuristic approach suits better for tasks with simpler rationales. Finally, dropping RNN can help the model parallelize and also less prone to mode collapse and gradient vanishing.

#### 4.5.2 Axillary Information

The improvement in rationale interpretability comes from regularizing the Generator through a wide set of knowledge sources, like the Relation, the Corpus and the Rationale heuristic, and we believe this can be further improved by including more knowledge sources. For example, the Entity heuristic can be improved by annotating entities through a large open source data such as Wikipedia, or the Corpus heuristic can be more diverse with distant supervision tricks. However, the amount of axillary information to include and to what degree is a challenge in itself.

Another major difference is that the Generator is directly related to the relation-labels through the Relation and Word heuristics, which is different from the original rationale framework where the Generator is not aware of the ground-truth. This can be a potential challenge when the diversity of relation vastly increases. To be more specific, the number of relations in some knowledge graph population tasks can be more than a few thousands, which can lead to severe label imbalance issues. Also, we notice several relation classes share the same rationales, like ‘of’ is commonly seen in Member-Collection and Component-Whole relations. This can be unfavorable for the Word heuristic since our approach assumes rationales are more heterogeneous than homogeneous between different relations. A potential solution is to cluster or introduce hierarchical structures to the relations thereby reducing diversity.

**Table 4.5** List of words/phrases that are selected as rationales in test set by our model. Additional words are denoted as ... if they are located between entities and rationales but are not selected as rationales.

Relation Class	Selected rationales with target entities
Cause-Effect	dips <sub>e1</sub> caused by ... y-rays <sub>e2</sub> , liver <sub>e1</sub> ... cause ... hypertension <sub>e2</sub> plaintiffs <sub>e1</sub> ...resulting from ... explosion <sub>e2</sub> , storm <sub>e1</sub> generated by ... cold <sub>e2</sub>
Component-Whole	site <sub>e1</sub> is part of ... network <sub>e2</sub> , cabinet <sub>e1</sub> encloses ... woofer <sub>e2</sub> , ladder <sub>e1</sub> comprises ... steps <sub>e2</sub> , crocodile <sub>e1</sub> has ... snout <sub>e2</sub>
Content-Container	guns <sub>e1</sub> are locked in safe <sub>e2</sub> , grenade <sub>e1</sub> hidden inside canister <sub>e2</sub> spider <sub>e1</sub> was contained in ... box <sub>e2</sub> , suitcase <sub>e1</sub> full ... books <sub>e2</sub>
Entity-Destination	weapons <sub>e1</sub> ... delivered to ... navy <sub>e2</sub> , money <sub>e1</sub> ... into ... custody <sub>e1</sub> children <sub>e1</sub> were handed over to relatives <sub>e2</sub> , water <sub>e1</sub> ... been poured into ... river <sub>e2</sub>
Entity-Origin	squirrel <sub>e1</sub> popped out of ... shirt <sub>e2</sub> , robbers <sub>e1</sub> ... away from scene <sub>e2</sub> gases <sub>e1</sub> emanated from ... sources <sub>e2</sub> , starch <sub>e1</sub> is source of sugars <sub>e2</sub>
Instrument-Agency	mechanic <sub>e1</sub> tightens ... with spanner <sub>e2</sub> , intellect <sub>e1</sub> wields pen <sub>e2</sub> project <sub>e1</sub> uses art <sub>e2</sub> as instrument, gives ... best practices <sub>e1</sub> for programmers <sub>e2</sub>
Member-Collection	bloat <sub>e1</sub> of hippopotamuses <sub>e2</sub> , formation <sub>e1</sub> comprised of ... dragoons <sub>e2</sub> surgeon <sub>e1</sub> is part of the team <sub>e2</sub> , sergeant <sub>e1</sub> in army <sub>e2</sub>
Message-Topic	caveats <sub>e1</sub> outlined ... e-mail <sub>e2</sub> , speech <sub>e1</sub> ... about conversation <sub>e2</sub> speech <sub>e1</sub> was summary of ... problems <sub>e2</sub> , parameters <sub>e1</sub> described text <sub>e2</sub>
Product-Producer	production materials <sub>e1</sub> by industries <sub>e2</sub> , products <sub>e1</sub> grown by ... defendant <sub>e2</sub> engineers <sub>e1</sub> ... devised ... method <sub>e2</sub> , investment firm <sub>e1</sub> co-founded by ... head <sub>e2</sub>

### 4.5.3 Potential Applications: Noise-Reduction in Distant Supervision

A potential application of this work, besides classifying relations and providing interpretable results, is to help reduce noise in distant supervision for relation classification. Distant supervision is a commonly used approach in relation classification where producing ground-truth data is expensive. Distant supervision exploits a simple assumption: if a sentence  $s$  contains an entity-relation pair  $(e1, relation, e2)$  that exists in a trustworthy knowledge source, then  $s$  can be a training data for the *relation*. As a result, distant supervision often contains many noisy false-negative training samples and degrades the result. To reduce the noise, one can penalize sentences that do not contain commonly used rationales for the relation.

## 4.6 Conclusion

The major difference between this work and the previous chapter is that we designed the Generator without a sequential RNN model. Specifically, we developed a layer by layer process that considers different levels of text information for three reasons. First, in relation extraction, the excess sequential modeling power from RNN is not required since rationales are often an extremely small span of words. Also, rationales are often strongly-related to POS tags which can be used to replace the sequence information. Second, the purpose of rationales in relation classification is to augment features and illuminate how the model derives the answer. This is different from other tasks like aspect-level sentiment classification in [Lei16] where the rationales are treated as a compression of the input.

Empirical results shown that this approach can improve results on both relation classification and rationale finding. In our experiments, relation classification performance is improved by 1.2 F1 score. Furthermore, rationale finding has a 20.0 improvement on F1 score, which pushes F1 score above 50.0. By analyzing several example sentences, we also see a better consistency of rationale selected. Finally, by dropping RNN from the original rationale framework, the model can now be easily parallelizable which leads to better performance. Another benefit is the model became less prone to mode collapse and gradient vanishing.

# CONCLUSION AND FUTURE WORK

## 5.1 Conclusion

Relation extraction is the process of converting information from text to connected components of interest that can be represented in a graph-like structure. In addition to providing a high-level view of desired knowledge, relation extraction also allows text information to be processed through graph algorithms. Due to the rapid improvements of neural network and text representation models, a fully autonomous end-to-end neural network based relation extraction model can easily outperform previous approaches that requires hand-crafted features. However, as neural network model has been long criticized as black boxes, these improvements came at the cost of interpretability and justifiability. To better understand the need of justifiability, one can imagine a scenario where a small portion of predicted relations are considered erroneous in a large scale knowledge graph system. Rather than adding more data and retrain the model, a more preferable way is to analyze and solve the root of the error with a human touch.

This dissertation has presented a novel rationale oriented framework for neural network based relation extraction to improve interpretability. **Rationales**, as defined in our work, are keyword(s) that a human reader would use to identify the embedded relation between the target entities in the sentence. This proposed framework performs relation extraction through a two-step approach: 1) identify rationales and 2) predict relations. The major benefit of using this proposed framework comes from the rationale identification step which provides justifiable results along with the pre-

dicted relation. Furthermore, the extracted rationales provide a opportunity for active-learning in the system. Human users can examine the selected rationales and correct the rationales to adjust the predicted relation. This allows a neural network based system to be actively open to modification and more interpretable.

While this dissertation is dedicated to the rationale oriented framework for relation extraction as a comprehensive work, each chapter can be used in a stand-alone fashion and are all published work in different areas. We reiterate the contributions of each chapter as follows:

In Chapter 2, we proposed a sentence classification model that examines both long-term, sequential word information and local phrase information to handle noisy sentences from on-line reviews like Yelp or IMDB where a grammar parser might fail. This model first uses a Recurrent Neural Network to analyze the sequential semantic development of the sentence, and at the meantime uses a Convolutional Neural Network to extract phrase information out of neighboring words. In order to combine long-term and local information, we used an attention based approach where each phrase information are weighted prior adding into the long-term memory. To make the attention mechanism adaptable to different sentence compositions, we used another Recurrent Neural Network to learn the weights and combine different information. Empirical results showed comparable results to state-of-the-art models on well-structured sentences where grammar parser works, and showed best-performing results on noisy sentences. Further analysis on different models showed that the attention mechanism is critical to the success of this hybrid approach.

In Chapter 3, we proposed the framework for the rationale-based relation extraction. This work is intended to extract rationales first and then utilize the rationales to make relation prediction. The major challenge of this approach is the lack of ground-truths for rationales. We implemented an generative adversarial network to facilitate a unsupervised module for rationale identification and a supervised module for relation extraction. Experiments showed improved results on relation classification accuracy comparing to state-of-the-art models, along with rationales that makes sense to human readers which are not available in previous works. In addition, we experimented supervised rationale identification using naive rationales as ground-truths, and showed the model is capable to correct the predilections with extra information is provided.

Finally, in Chapter 4, we attempt to address the mode-collapse issue in the rationale based relation extraction framework. Mode-collapse happens when the model attempted and failed to learn good rationales through various signals, i.e. word information, sentence composition, Part-of-Speech tags and etc, and results to learn only from single source of information. We provided another approach for learning rationales by using multi-level information obtainable from the sentence. To validate the quality of rationales of this alternative approach, we labeled the rationales in the test set of the dataset with three human annotators. Experiments showed major improvement on the rationales quality comparing to the original method.

## 5.2 Future Work

While this dissertation provides a novel approach for relation extraction, it shows several many new research directions for the area. To begin with, while in this work rationales are mainly discovered using unsupervised methods, we are interested if labeled-data are to be provided. While creating a training set for rationales can be cumbersome, we propose future research can first utilize this existing framework to generate rationales on a small set of sentences and build up the labeled dataset by correcting these rationales later. Another approach, which is not mutually exclusive to the one just mentioned, is to manually create a small set of rationales for each relation with a group of annotators and use semi-supervised training to expand the rationale set. We believe both approach can be taken at th same time and be perfect by active human annotators involved to create the labeled rationale dataset. Once a labeled rationale dataset is created, one can then design a method to combine the supervised and unsupervised results and measure the improvements. The final goal would be creating a justifiable knowledge-graph system.

Another future research direction is how to apply this model to large knowledge-graph systems where there are massive relations available. As types of relations expanded, it requires more data to learn rationales and can cause mode-collapse issue when the dataset is unbalanced in the number of relation labels. A potential approach is to create a ontology of relation types that helps to build structure within relation types and hence de-noises the system. These structures can be either manually curated or be learned from large cataloging sources like Wikipedia.

For most NLP related research, a natural future work is to generalize the model to support different languages. We hypothesize this rationale-based framework would work on most language as it works without grammar or structural information. However, a potential issue is that different language might have rationales of a different format, or might have a massive variety of words to describe relationship and hence word heuristics would be less effective.

Finally, while this framework is designed for sentence level relation extraction, another interesting direction to explore is to generalize this framework to document and discourse level relation extraction. Document level relation extraction aims to extract relation that lapse cross sentences that draws increasing interest to better cover relation information that are not fully expressed in one sentence[YM16; QP16; Gu16; Pen17]. We use an example that was used in [Pen17] in the following to demonstrate cross sentence relation where **exon-19** ,**EGFR** and **gefitinib** has a tenery relation that are not fully expressed in either sentence:

*“The deletion mutation on **exon-19** of **EGFR** gene was present in 16 patients, while the L858E point mutation on exon-21 was noted in 10. All patients were treated with **gefitinib** and showed a partial response.”*

With the increased complexity, a potential solution would be to include structural information within and between sentences to better guide rationale selection. There are several methods proposed in machine summarization area to help build such structure and can be utilized for this task[Bel15; Liu18; Kik14]. Another challenge that can be expected is as we approach to document level relation extraction, the form of rationales might need to be extended from a small collection of words in the sentence to a more complex format. For example words scattered in multiple sentences. As the complexity of the form of the rationales increases, the major benefit of rationales - interpretability, can diminish and proposed another challenge worthy to be explored.

## BIBLIOGRAPHY

- [Ang15] Angeli, G. et al. “Leveraging linguistic structure for open domain information extraction”. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Vol. 1. 2015, pp. 344–354.
- [AB17] Arjovsky, M. & Bottou, L. “Towards principled methods for training generative adversarial networks.” *arXiv preprint arXiv:1701.04862*. 2017.
- [Arj17] Arjovsky, M. et al. “Wasserstein gan”. *arXiv preprint arXiv:1701.07875, 2017*. 2017.
- [BB11] Bach, N. & Badaskar, S. “A Review of Relation Extraction” (2011).
- [Bah15] Bahdanau, D. et al. “Neural machine translation by jointly learning to align and translate”. *ICLR 2015*. San Diego, California, 2015.
- [Bel15] Beliga, S. et al. “An overview of graph-based keyword extraction methods and approaches”. *Journal of information and organizational sciences* **39**.1 (2015), pp. 1–20.
- [Che17] Che, T. et al. “Mode Regularized Generative Adversarial Networks”. *ICLR*. 2017.
- [Che16] Chen, X. et al. “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets”. *NIPS*. 2016.
- [Cho14] Cho, K. et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734.
- [Chu14] Chung, J. et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. *Advances in Neural Information Processing Systems(NIPS) Workshop* (2014).
- [CM16] Clark, K. & Manning, C. D. “Improving coreference resolution by learning entity-level distributed representations”. *arXiv preprint arXiv:1606.01323* (2016).
- [Duc11] Duchi, J. et al. “Adaptive subgradient methods for online learning and stochastic optimization”. *Journal of Machine Learning Research* **12**.Jul (2011), pp. 2121–2159.
- [DH18] Duhyeon, B. & Hyunjung, S. “Improved Training of Generative Adversarial Networks Using Representative Features”. *arxiv: preprint arXiv:1801.09195*. 2018.
- [Dum09] Dumitru, E. et al. “Visualizing higher-layer features of a deep network”. *University of Montreal 1341*. Vol. 3. 2009.

- [Eic08] Eichler, K. et al. “Unsupervised relation extraction from web documents”. *LREC 2008* (2008).
- [Goo14] Goodfellow, I. et al. “Generative adversarial nets”. *NIPS*. 2014, pp. 2672–2680.
- [Gor15] Gormley, M. R. et al. “Improved relation extraction with feature-rich compositional embedding models”. *EMNLP*. 2015, pp. 1774–1784.
- [Gra13] Graves, A. “Generating sequences with recurrent neural networks”. *arXiv preprint arXiv:1308.0850* (2013).
- [Gu16] Gu, J. et al. “Chemical-induced disease relation extraction with various linguistic features”. *Database* **2016** (2016).
- [Gul17] Gulrajani, I. et al. “Improved training of wasserstein gans.” *arXiv preprint arXiv:1704.00028*. 2017.
- [Hen09] Hendrickx, I. et al. “Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals”. *NAACL HLT*. 2009, pp. 94–99.
- [HS97] Hochreiter, S. & Schmidhuber, J. “Long short-term memory”. *Neural computation* **9.8** (1997), pp. 1735–1780.
- [Hsu18] Hsu, S. T. et al. “An Interpretable Generative Adversarial Approach to Classification of Latent Entity Relations from Unstructured Sentences”. *AAAI*. 2018.
- [IC14] Irsoy, O. & Cardie, C. “Deep recursive neural networks for compositionality in language”. *Advances in Neural Information Processing Systems*. Montreal, Canada, 2014, pp. 2096–2104.
- [Kal14a] Kalchbrenner, N. et al. “A Convolutional Neural Network for Modelling Sentences”. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, 2014, pp. 655–665.
- [Kal14b] Kalchbrenner, N. et al. “A convolutional neural network for modelling sentences”. *ACL*. 2014, pp. 655–665.
- [Kik14] Kikuchi, Y. et al. “Single document summarization based on nested tree structure”. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vol. 2. 2014, pp. 315–320.
- [Kim14a] Kim, Y. “Convolutional Neural Networks for Sentence Classification”. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1746–1751.

- [Kim14b] Kim, Y. “Convolutional neural networks for sentence classification”. *EMNLP*. 2014, pp. 1746–1751.
- [KB14] Kingma, D. P. & Ba, J. “Adam: A method for stochastic optimization”. *arXiv preprint arXiv:1412.6980* (2014).
- [Kod17] Kodali, N. et al. “On convergence and stability of gans”. *arXiv preprint arXiv:1705.07215*. 2017.
- [Kri12] Krizhevsky, A. et al. “Imagenet classification with deep convolutional neural networks”. *Advances in neural information processing systems*. Lake Tahoe, Nevada, 2012, pp. 1097–1105.
- [LM14] Le, Q. V. & Mikolov, T. “Distributed Representations of Sentences and Documents.” *Proceedings of The 31st International Conference on Machine Learning*. Beijing, China, 2014, pp. 1188–1196.
- [LeC98] LeCun, Y. et al. “Gradient-based learning applied to document recognition”. *Proceedings of the IEEE* **86**.11 (1998), pp. 2278–2324.
- [Lei16] Lei, T. et al. “Rationalizing neural predictions”. *EMNLP*. 2016, pp. 107–117.
- [Liu18] Liu, F. et al. “Toward abstractive summarization using semantic representations”. *arXiv preprint arXiv:1805.10399* (2018).
- [Liu15a] Liu, Y. et al. “A dependency-based neural network for relation classification”. *ACL*. 2015, pp. 285–290.
- [Liu15b] Liu, Y. et al. “Topical Word Embeddings.” 2015.
- [Mac15] Maclaurin, D. et al. “Gradient-based hyperparameter optimization through reversible learning”. *International Conference on Machine Learning*. 2015, pp. 2113–2122.
- [Mar13] Martin, M. K. et al. “Network text analysis of conceptual overlap in interviews, newspaper articles and keywords”. *Social Network Analysis and Mining* **3.4** (2013), pp. 1165–1177.
- [Met16] Metz, L. et al. “Unrolled generative adversarial networks”. *arXiv preprint arXiv:1611.02163* (2016).
- [Mik13] Mikolov, T. et al. “Distributed representations of words and phrases and their compositionality”. *Advances in Neural Information Processing Systems*. Lake Tahoe, Nevada, 2013, pp. 3111–3119.
- [MO14] Mirza, M. & Osindero, S. “Conditional generative adversarial nets”. *arXiv preprint arXiv:1411.1784* (2014).

- [MB16] Miwa, M. & Bansal, M. “End-to-end relation extraction using lstms on sequences and tree structures”. *ACL*. 2016, pp. 1105–1116.
- [Moe06] Moens, M.-F. *Information extraction: algorithms and prospects in a retrieval context*. Vol. 21. Springer Science & Business Media, 2006.
- [Mon17] Montavon, G. et al. “Methods for interpreting and understanding deep neural networks”. *CoRR*. [abs/1706.07979](https://arxiv.org/abs/1706.07979). 2017.
- [Mou14] Mou, L. et al. “TBCNN: A tree-based convolutional neural network for programming language processing”. *CoRR*, [abs/1409.5718](https://arxiv.org/abs/1409.5718) (2014).
- [Mou15] Mou, L. et al. “Discriminative Neural Sentence Modeling by Tree-Based Convolution”. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 2315–2325.
- [Muz15] Muzaffar, A. W. et al. “A relation extraction framework for biomedical text using hybrid feature set”. *Computational and mathematical methods in medicine* **2015** (2015).
- [Nes] Nesterov, Y. “A method of solving a convex programming problem with convergence rate  $O(1/k^2)$ ”.
- [Ngu16] Nguyen, A. et al. “Synthesizing the preferred inputs for neurons in neural networks via deep generator networks”. *NIPS*. 2016, pp. 3387–3395.
- [NG15a] Nguyen, T. H. & Grishman, R. “Combining neural networks and log-linear models to improve relation extraction”. *IJCAI Workshop on Deep Learning for Artificial Intelligence (DLAI)*. 2015.
- [NG15b] Nguyen, T. H. & Grishman, R. “Relation extraction: Perspective from convolutional neural networks”. *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. 2015, pp. 39–48.
- [Niu12] Niu, F. et al. “DeepDive: Web-scale Knowledge-base Construction using Statistical Learning and Inference.” *VLDS* **12** (2012), pp. 25–28.
- [Pal16] Palangi, H. et al. “Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval”. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* **24.4** (2016), pp. 694–707.
- [Pen17] Peng, N. et al. “Cross-sentence n-ary relation extraction with graph lstms”. *arXiv preprint arXiv:1708.03743* (2017).
- [Pha14] Pham, V. et al. “Dropout improves recurrent neural networks for handwriting recognition”. *Proceedings International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Crete, Greece, 2014, pp. 285–290.

- [Qia99] Qian, N. “On the momentum term in gradient descent learning algorithms”. *Neural networks* **12.1** (1999), pp. 145–151.
- [QP16] Quirk, C. & Poon, H. “Distant supervision for relation extraction beyond the sentence boundary”. *arXiv preprint arXiv:1609.04873* (2016).
- [RR09] Ratnoff, L. & Roth, D. “Design challenges and misconceptions in named entity recognition”. *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics. 2009, pp. 147–155.
- [Red18] Reddi, S. J. et al. “On the Convergence of Adam and Beyond”. *International Conference on Learning Representations*. 2018.
- [Ree16] Reed, S. et al. “Generative adversarial text to image synthesis”. *arXiv preprint arXiv:1605.05396* (2016).
- [RH10] Rink, B. & Harabagiu, S. “Utd: Classifying semantic relations by combining lexical and semantic resources”. *SemEval*. 2010, pp. 256–259.
- [Rot17] Roth, K. et al. “Stabilizing Training of Generative Adversarial Networks through Regularization.” *NIPS*. 2017.
- [Rus15] Rush, A. M. et al. “A neural attention model for abstractive sentence summarization”. *arXiv preprint arXiv:1509.00685* (2015).
- [Sah16] Sahu, S. K. et al. “Relation extraction from clinical texts using domain invariant convolutional neural network”. *arXiv preprint arXiv:1606.09370* (2016).
- [Sal16] Salimans, T. et al. “Improved techniques for training gans”. *Advances in Neural Information Processing Systems (NIPS)*. 2016, pp. 2234–2242.
- [Sam16] Samek, W. et al. “Evaluating the visualization of what a deep neural network has learned.” *IEEE transactions on neural networks and learning systems* **28** (2016), pp. 2660–2673.
- [Sar08] Sarawagi, S. et al. “Information extraction”. *Foundations and Trends in Databases* **1.3** (2008), pp. 261–377.
- [See17] See, A. et al. “Get to the point: Summarization with pointer-generator networks”. *arXiv preprint arXiv:1704.04368* (2017).
- [Sim13] Simonyan, K. et al. “Deep inside convolutional networks: Visualising image classification models and saliency maps”. *CoRR. abs/:1312.6034* (2013).
- [Soc11] Socher, R. et al. “Parsing natural scenes and natural language with recursive neural networks”. *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 129–136.

- [Soc12a] Socher, R. et al. “Semantic compositionality through recursive matrix-vector spaces”. *EMNLP*. 2012, pp. 1201–1211.
- [Soc12b] Socher, R. et al. “Semantic Compositionality through Recursive Matrix-Vector Spaces”. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea: Association for Computational Linguistics, 2012, pp. 1201–1211.
- [Soc13] Socher, R. et al. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank”. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington: Association for Computational Linguistics, 2013, pp. 1631–1642.
- [Sri17] Srivastava, A. et al. “VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning.” *NIPS*. 2017.
- [Sut13] Sutskever, I. “Training recurrent neural networks” (2013).
- [Sut14] Sutskever, I. et al. “Sequence to sequence learning with neural networks”. *Advances in Neural Information Processing Systems*. Montreal, Canada, 2014, pp. 3104–3112.
- [Tai15] Tai, K. S. et al. “Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks”. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, 2015, pp. 1556–1566.
- [Tan15] Tang, D. et al. “Document Modeling with Gated Recurrent Neural Network for Sentiment Classification”. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 1422–1432.
- [Tul17] Tulyakov, S. et al. “Mocogan: Decomposing motion and content for video generation”. *arXiv preprint arXiv:1707.04993* (2017).
- [Wan16] Wang, L. et al. “Relation Classification via Multi-Level Attention CNNs”. *ACL*. 2016, pp. 1298–1307.
- [Wu17] Wu, T. et al. “Interpretable R-CNN”. *arXiv preprint arXiv:1711.05226* (2017).
- [Xu15] Xu, K. et al. “Semantic relation classification via convolutional neural networks with simple negative sampling”. *EMNLP*. 2015, pp. 536–540.
- [Xu17] Xu, M. et al. “A Lifelong Learning Topic Model Structured Using Latent Embeddings”. *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*. 2017, pp. 260–261.

- [Yam17] Yamamoto, A. et al. “Company Relation Extraction from Web News Articles for Analyzing Industry Structure”. *Semantic Computing (ICSC), 2017 IEEE 11th International Conference on*. IEEE. 2017, pp. 89–92.
- [YM16] Yang, B. & Mitchell, T. “Joint extraction of events and entities within a document context”. *arXiv preprint arXiv:1609.03632* (2016).
- [Yan16] Yang, Z. et al. “Hierarchical attention networks for document classification”. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, pp. 1480–1489.
- [Yi17] Yi, Z. et al. “DualGAN: Unsupervised Dual Learning for Image-to-Image Translation.” *ICCV*. 2017, pp. 2868–2876.
- [YS15] Yin, W. & Schütze, H. “Multichannel Variable-Size Convolution for Sentence Classification”. *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Beijing, China: Association for Computational Linguistics, 2015, pp. 204–214.
- [Yu14a] Yu, L. et al. “Deep Learning for Answer Sentence Selection”. *NIPS Deep Learning Workshop*. Montreal, Canada, 2014.
- [Yu14b] Yu, M. et al. “Factor-based compositional embedding models”. *NIPS*. 2014, pp. 95–101.
- [Zai07] Zaidan, O. et al. “Using Annotator Rationales to Improve Machine Learning for Text Categorization”. *NAACL HLT*. 2007, pp. 260–267.
- [Zei12] Zeiler, M. D. “ADADELTA: an adaptive learning rate method”. *CoRR, abs/1212.5701* (2012).
- [Zen14] Zeng, D. et al. “Relation Classification via Convolutional Deep Neural Network”. *COLING*. 2014, pp. 2335–2344.
- [Zen15] Zeng, D. et al. “Distant supervision for relation extraction via piecewise convolutional neural networks”. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 1753–1762.
- [Zha16a] Zhang, Y. et al. “MGNC-CNN: A Simple Approach to Exploiting Multiple Word Embeddings for Sentence Classification”. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, 2016, pp. 1522–1527.
- [Zha16b] Zhang, Y. et al. “Rationale-augmented convolutional neural networks for text classification”. *EMNLP*. 2016, pp. 795–804.
- [Zhu17] Zhu, J.-Y. et al. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. *arXiv preprint* (2017).

[Zhu15] Zhu, X.-D. et al. “Long Short-Term Memory Over Recursive Structures.” *ICML*. Lille, France, 2015, pp. 1604–1612.