The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

An Interpretable Generative Adversarial Approach to Classification of Latent Entity Relations in Unstructured Sentences

Shiou Tian Hsu, Changsung Moon, Paul Jones, Nagiza F. Samatova
North Carolina State University, Raleigh, NC, 27606
{shsu3,cmoon2,pjones}@ncsu.edu, samatova@csc.ncsu.edu

Abstract

We propose a generative adversarial neural network model for relation classification that attempts to emulate the way in which human analysts might process sentences. Our approach provides two unique benefits over existing capabilities: (1) we make predictions by finding and exploiting supportive rationales to improve interpretability (i.e. words or phrases extracted from a sentence that a person can reason upon), and (2) we allow predictions to be easily corrected by adjusting the rationales. Our model 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 through human annotations. We evaluate our model on data from SemEval 2010 that provides 19 relation-classes. Experiments demonstrate that our approach outperforms state-of-the-art models, and that our model is capable of extracting good rationales on its own as well as benefiting from labeled rationales if provided.

Introduction

Sentence-level entity relation classification is the task of recognizing the relationship between two entities within a sentence. It is the core function of autonomous knowledge discovery in unstructured text (such as web documents), and thus is critical for applications including information extraction, knowledge base construction, and many other higher level NLP tasks (**?**; **?**). 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 classification has greatly benefited from recent developments in neural network models, and these benefits have propagated into many other important applications like document classification, sentiment analysis, summarization, topic discovery, word-sense disambiguation, co-reference identification and more. Initially neural models pushed classification performance to new heights, improving F1 score on relation classification to 88% (**?**) compared to 82.2% using an SVM with engineered features (**?**). 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*$_{0.02}$ *disgusting*$_{0.03}$ *scene*$_{0.1}$ *was*$_{0.02}$ *retaliation*$_{0.03}$ *against*$_{0.07}$ *her*$_{0.015}$ *brother*$_{0.02}$ *Philip*$_{0.04}$ *who*$_{0.08}$ *rents*$_{0.03}$ *the*$_{0.1}$ *[room]*e_1 *inside*$_{0.13}$ *this*$_{0.03}$ *apartment*$_{0.09}$ *[house]*e_2 *on*$_{0.01}$ *Lombard*$_{0.03}$ *street*$_{0.08}$.
(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 (**?**; **?**; **?**). 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

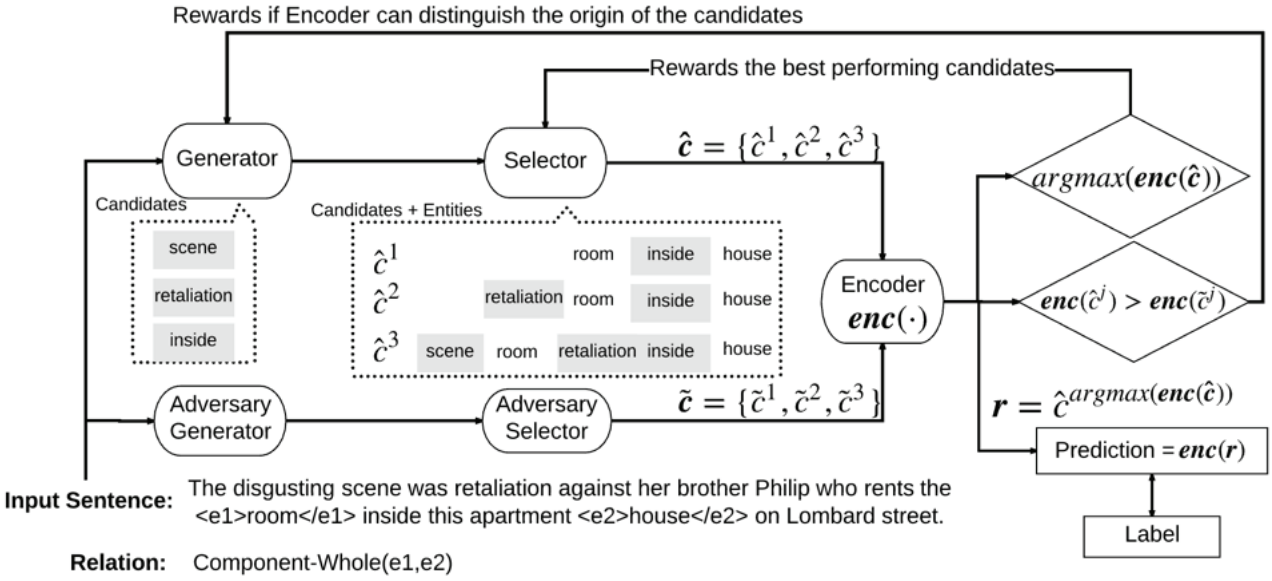

Figure 1: Model Structure with sample input sentence. Entities are given from the dataset. The Generator selects candidate rationales, and the Selector enumerates all possible combinations of candidates with entities and selects one 'best performing candidate set'. An adversary Generator and Selector 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.

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 **??**. 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.

Experiments demonstrate that our model produces an 89.5% F1 score on relation classification tasks, compared to a score of 88.0% for the leading state-of-the-art approach (**?**). While the quality of interpretability is difficult to measure without human evaluation, we make the assumption that rationales that lead to better F1 scores are more interpretable than those that lead to lower scores. In effect, we treat the Encoder as a proxy human reader. Based on this assumption, we evaluated the quality of our rationales by treating attention models as proxy rationale generating models, where rationales are the highest-scoring consecutive n-grams in the attention model. Compared to the proxy rationale models, experiments using rationales generated from our model show a 1.7∼5.0% improvement in F1 score. In addition, in-depth analysis shows that our model is capable of identifying good rationales that a human reader can use to infer relationships, and predictions can be corrected if provided with better rationales. We naively annotated some rationales using the guidelines in **?** (**?**). We discovered that, even with minimal annotations on the dataset, the refined results showed improvement in both F1 score and training speed.

Related Work

Interpretable neural network models have recently become of increased interest to the natural language processing research community (**?**). Several approaches to interpretability have been introduced, including: visualization of activation-functions within neural networks (**?**; **?**), accrediting distributed word vectors (**?**), relevance propagation (**?**), attention based models (**?**), and rationale finding (**?**; **?**; **?**). Even though each model is optimized for a slightly different task, a common approach they adopted is to discriminate the input data and selectively use fragments from the input. Interpretability can then be measured by the quality of the selected fragments.

The major distinction between the attempts to demystify neural networks is thus the degree of discrimination imposed on the input data. Less-discriminating approaches like attention based models usually allow the model to use the whole input data. These models introduce weights into the models where the weights are commonly generated based on a sim-

ilarity function between signals in the sentence with the desired goal. Signals, which are often n-grams in the sentence, receive higher weights if analogous to the desired goal, and can be considered as "highlights" of the data. While the major benefit of this approach is that can be effective even with naive methods like n-gram to generate signals, weights attributed to the signals do not always correspond to how humans might approach a given problem. Also, complexity of signal generation can become exponential if considering *nonconsecutive* n-grams cases. Most recent research in entity-relation classification falls into the low-constraint category.

High-discriminating approaches, which aim to extract rationales for reasoning, pose a higher degree of constraint of selecting data to be used in the model. Models that fall under this category are usually trained to select only part of the data for reasoning, and to obscure the rest of the data. 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. Furthermore, it introduces methods to selectively ignore data to be considered, which can in turn helps downsize complexity for skipping n-grams cases.

We position this work on the high-constraint (rationale-finding) side comparing with the attention-based models. In our approach, we find and use nonconsecutive words as rationales in a completely unsupervised way in $O(NM)$ where $M \leq N$.

Adversarial Rationale Generation

We formalize here the general idea of rationale generation for entity relationship classification in this research. Consider an input sequence of words $\boldsymbol{x} = \{x_t\}_{t=1}^{T}$, where e_1 and e_2 are the entities of interest and e_1 and $e_2 \subset \{x_1, x_2, ..., x_T\}$. T is the length of the input sequence. The goal of relation classification is to use ($\boldsymbol{x}$,e_1,e_2) to predict the relationship **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 ($\boldsymbol{x}$,e_1,e_2) to a target vector that is representative of the desired relationship. We use $\boldsymbol{enc}(\boldsymbol{x},e_1,e_2)$ 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 classification is to extract a set of rationales $\boldsymbol{r}$ that are incorporated into the encoding process as $\boldsymbol{enc}(\boldsymbol{r}, \boldsymbol{x},e_1,e_2)$, where $\boldsymbol{r} \subset \{x_1, x_2, ..., x_T\}$ but $\boldsymbol{r} \notin \{e_1, e_2\}$. 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 $\boldsymbol{gen}(\boldsymbol{x},e_1,e_2)$, which samples $\boldsymbol{c} = \{c_t\}_{t=1}^{T}$ 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 $\boldsymbol{r}$ from $\boldsymbol{c}$ and is represented as $\boldsymbol{sel}(\boldsymbol{c},e_1,e_2)$.

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 $\boldsymbol{enc}(\boldsymbol{r},\boldsymbol{x},e_1,e_2)$ can outperforms another set of rationales $\boldsymbol{enc}(\tilde{\boldsymbol{r}},\boldsymbol{x},e_1,e_2)$.

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 $\boldsymbol{r_{re}}$ 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 **?** (**?**), the aim is to generate concise and reasonable summaries for customer reviews. In their research, the goal of $\boldsymbol{gen}(\boldsymbol{x})$ is to generate rationales $\boldsymbol{r}$ that ensure that $\boldsymbol{enc}(\boldsymbol{x})$ and $\boldsymbol{enc}(\boldsymbol{r})$ are at similar points in the target vector space. In other words, the goal of $\boldsymbol{r}$ in their research is to serve as a proxy of $\boldsymbol{x}$ and fool $\boldsymbol{enc}(\cdot)$, while in our work the goal of $\boldsymbol{r}$ is to outperform all possible short enumerations of $\boldsymbol{x}$ in $\boldsymbol{enc}(\cdot)$.

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.

Encoder

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

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

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 ($\boldsymbol{r}$,$\boldsymbol{x}$,e_1,e_2), and 2) it generates $\hat{y}$ by projecting the representation of ($\boldsymbol{r}$,$\boldsymbol{x}$,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 rationals $\boldsymbol{r}$ 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 (**?**; **?**). As in **?** (**?**), the CNN model contains a set of convolutional filters W^ℓ and bias terms b^ℓ. These filters are applied to all possible consecutive n-grams of length ℓ 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 $\boldsymbol{r}$ = $\{x_{10}, x_{15}\}$, $e_1 = x_4$ 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 $\boldsymbol{r}$ = $\{x_3, x_{10}\}$.

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

$$
\begin{aligned}
&\hat{y} = \boldsymbol{enc}(\boldsymbol{r}, \boldsymbol{x}, e_1, e_2) \\
&= softmax(W_{enc} \cdot [V_r \oplus V_x \oplus e_1 \oplus e_2] + b_{enc}) \\
&\quad (\oplus = \text{vector concatenation}) \\
&V_r = Relu(W_r \cdot [\boldsymbol{r}, e_1, e_2] + b_r) \\
&V_{x_t}^\ell = Relu(W_x^\ell \cdot x_{t:t+\ell} + b_x^\ell) \\
&V_x^\ell = [V_{x_1}^\ell, V_{x_2}^\ell, ..., V_{x_{T-\ell}}^\ell] \\
&V_x = \{maxpooling(V_x^\ell)\}^{\ell \in L}
\end{aligned}
\tag{2}
$$

Generator

The goal of the Generator $\boldsymbol{gen}(\boldsymbol{x})$ is to extract pieces of text from input sentence $\boldsymbol{x}$ 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 $\boldsymbol{c} = \{c_t\}_{t=1}^T$ where each $c_t \in [0, 1]$ is whether x_t is chosen as a candidate rationale. We represent it as $\boldsymbol{c} \sim \boldsymbol{gen}(\boldsymbol{x})$.

We adopted a recurrent neural network (RNN) model in this work to compute $\boldsymbol{c}$ due to the sequential characteristics of $\boldsymbol{x}$ and $\boldsymbol{c}$. 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.

$$
\begin{aligned}
&RecurrentUnit: \\
&\quad z_t = \sigma(W_z \cdot x_t + U_z \cdot h_{t-1} + b_z) \\
&\quad o_t = \sigma(W_r \cdot x_t + U_r \cdot h_{t-1} + b_r) \\
&\quad \hat{h}_t = \tanh(W_h \cdot x_t + U_h \cdot (h_{t-1} \circ o_t) + b_h) \\
&\quad h_t = z_t \circ \hat{h}_t + (1 - z_t) \circ h_{t-1} \\
&\qquad (\circ = \text{elementwise-multiplication}) \\
&\overleftarrow{h_t} = RecurrentUnit(x_t, \overleftarrow{h_{t+1}}) \\
&\overrightarrow{h_t} = RecurrentUnit(x_t, \overrightarrow{h_{t-1}}) \\
&s_t = sigmoid(W_s \cdot [\overleftarrow{h_t}, \overrightarrow{h_t}, e_1, e_2] + b_s)
\end{aligned}
\tag{3}
$$

In the most simplistic generator, the probability that x_t is chosen is conditionally independent of all other $\boldsymbol{x}$. In other words, we can sample candidate $\boldsymbol{c}$ 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 $\boldsymbol{c}$ to limit the number of rationales selected to be at most J.

$$
\begin{aligned}
&c_t = \begin{cases} 1, sort(\{(s_t > rand(0,1)) \cdot s_t\}_{t=1}^T)[1:J] \\ 0, otherwise \end{cases} \\
&\boldsymbol{c} = \{c_t\}_{t=1}^T
\end{aligned}
\tag{4}
$$

Selector

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

$$
\begin{aligned}
&\hat{c}_t^j = \begin{cases} 1, sort(\{c_t \cdot s_t\}_{t=1}^T)[1:j], j \in [1:J] \\ 0, otherwise \end{cases} \\
&\hat{\boldsymbol{c}}^{\boldsymbol{j}} = \{\hat{c}_t^j\}_{t=1}^T, \text{where } \hat{c}_t^j = 1 \\
&score(\hat{\boldsymbol{c}}^{\boldsymbol{j}}) = W_c \cdot \boldsymbol{enc}(\hat{\boldsymbol{c}}^{\boldsymbol{j}}, \boldsymbol{x}, e1, e2) \\
&scores^j = softmax(\{score(\hat{\boldsymbol{c}}^{\boldsymbol{j}})\}_{j=1}^J)^j \\
&\hat{scores}^j = \begin{cases} 1 & , j = argmax(scores) \\ 0 & , otherwise \end{cases} \\
&\boldsymbol{r} = \{x_t\}, \text{where } \hat{c}_t^j = 1,\ j = argmax(scores)
\end{aligned}
\tag{5}
$$

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.

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 (**?**). 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 $\tilde{\boldsymbol{c}}$ randomly from $\boldsymbol{x}$. The adversary rationales $\tilde{\boldsymbol{r}}$ will then be selected using $\tilde{\boldsymbol{c}}$ as per Equation (5). $\tilde{\boldsymbol{r}}$ 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 $j \neq argmax(scores)$ 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:

$$
\begin{aligned}
f(c_t^j) &= -log(s_t) \cdot c_t^j - log(1 - s_t) \cdot (1 - c_t^j) \\
f_y(\hat{y}_i^j) &= -y_i \cdot log(\hat{y}_i^j) \\
\mathcal{L}_{generator} &= \sum_{j=1}^{J} \sum_{t=1}^{T} \mathcal{P}^j \left[\mathcal{D}^j f(\hat{c}_t^j) + (1 - \mathcal{D}^j) f(\tilde{c}_t^j) \right] \\
\mathcal{L}_{selector} &= \sum_{j=1}^{J} -\mathcal{D}^j * \hat{scores}^j * log(scores^j) \\
\mathcal{L}_{encoder} &= \sum_{j=1}^{J} P^j \left[\mathcal{D}^j f_y(\hat{y}^j) - (1 - \mathcal{D}^j) f_y(\tilde{y}^j) \right]
\end{aligned}
\tag{6}
$$

The final joint objective and expected cost is defined as:

$$
\mathcal{L}(\boldsymbol{r}, \boldsymbol{x}, e_1, e_2, y) = \mathcal{L}_{generator} + \mathcal{L}_{selector} + \mathcal{L}_{encoder} \tag{7}
$$

$$
\min_{\theta e \theta g \theta s} \sum_{(\boldsymbol{x}, e_1, e_2, y) \in N} \mathbb{E}_{\boldsymbol{r} \sim sel(\boldsymbol{c}), \boldsymbol{c} \sim gen(\boldsymbol{x})} \mathcal{L}(\boldsymbol{r}, \boldsymbol{x}, e_1, e_2, y) \tag{8}
$$

where N is the training set and θe, θg, θs 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 $\tilde{\boldsymbol{r}}$ 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 $\boldsymbol{r_{re}}$ are provided, then we change $\boldsymbol{r}$ in Equation (2) and $\hat{\boldsymbol{c}}$ in Equation (6) to $\boldsymbol{r_{re}}$. This can be understood as a simple sequential discriminatory model where the Generator and the Selector is guided by $\boldsymbol{r_{re}}$. However, while keeping the reminder of the model the same, introducing $\boldsymbol{r_{re}}$ does not guarantee the Selector and Encoder will use $\boldsymbol{r_{re}}$ while reasoning.

Experiments

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.

Experimental Setup

Here we describe the detailed parameters used in the experiment. We initialized word vectors using the GoogleNews[1] 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 ℓ. 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 re-sampled 50 times when training the Selector and Encoder.

Results

We present the comparison between our model and other models in Table **??**. 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 (**?**) they used the whole dependency tree, while in SPTree (**?**) 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.

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

Classifier	F1
Manually Engineered Models	
SVM(**?**)	82.2
Dependency Tree Models	
RNN (**?**)	77.6
MVRNN (**?**)	82.4
FCM (**?**)	83.0
Hybrid FCM (**?**)	83.4
DepNN (**?**)	83.6
DRNNs (**?**)	85.6
SPTree (**?**)	84.5
Independent Models	
CNN & softmax (**?**)	82.7
Stackforward(**?**)	83.4
Vote-bidirect (**?**)	84.1
Vote-backward (**?**)	84.1
ATT-Input-CNN (**?**)	87.5
ATT-Pooling-CNN (**?**)	88.0
Our model	**89.5**

Table 1: Comparisons with benchmarking models

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 **?** (**?**), and added attention weights calculated by the similarity with ($e1$,$e2$). 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 **??**.

Classifier	F1
Proxy attention model	87.5
Proxy rationales + Encoder	
One rationale from proxy	84.5
Two rationales from proxy	85.3
Three rationales from proxy	87.8
Our model	**89.5**

Table 2: Comparison with proxy rationale models.

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

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 **??**. 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) converging, 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.

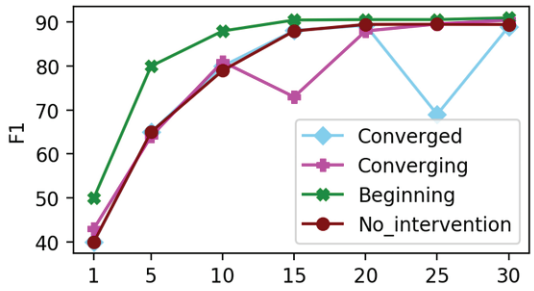

Figure 2: F1 results using different intervention strategy.

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)"*

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

Table 3: 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 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

Table 4: List of ground-truth rationales used

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."*

Discussion

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.

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.

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.

Potential Uses

Besides relation classification, another potential use-case for our model is in generating rules for relation classification. It should be possible to automatically discover sets of rules for relation classification 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.

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.

Acknowledgments

We thank the AAAI reviewers for their detailed and insightful feedback. This material is based upon work supported in whole or in part with funding from 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.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Faruqui, M.; Dodge, J.; Jauhar, S. K.; Dyer, C.; Hovy, E.; and Smith, N. A. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL HLT*, 1606–1615.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*, 2672–2680.

Gormley, M. R.; Yu, M.; and Dredze, M. 2015. Improved relation extraction with feature-rich compositional embedding models. In *EMNLP*, 1774–1784.

Hendrickx, I.; Kim, S. N.; Kozareva, Z.; Nakov, P.; Ó Séaghdha, D.; Padó, S.; Pennacchiotti, M.; Romano, L.; and Szpakowicz, S. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *NAACL HLT*, 94–99.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A convolutional neural network for modelling sentences. In *ACL*, 655–665.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, 1746–1751.

Lei, T.; Barzilay, R.; and Jaakkola, T. 2016. Rationalizing neural predictions. In *EMNLP*, 107–117.

Liu, Y.; Wei, F.; Li, S.; Ji, H.; Zhou, M.; and Wang, H. 2015. A dependency-based neural network for relation classification. In *ACL*, 285–290.

Miwa, M., and Bansal, M. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL*, 1105–1116.

Montavon, G.; Samek, W.; and Müller, K.-R. 2017. Methods for interpreting and understanding deep neural networks. In *CoRR. abs/1706.07979*.

Nguyen, T. H., and Grishman, R. 2015. Combining neural networks and log-linear models to improve relation extraction. In *IJCAI Workshop on Deep Learning for Artificial Intelligence (DLAI)*.

Nguyen, A.; Dosovitskiy, A.; Yosinski, J.; Brox, T.; and Clune, J. 2016. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *NIPS*, 3387–3395.

Niu, F.; Zhang, C.; Ré, C.; and Shavlik, J. W. 2012. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS* 12:25–28.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Knowledge Discovery and Data Mining(KDD)*, 1135–1144.

Rink, B., and Harabagiu, S. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *SemEval*, 256–259.

Samek, W.; Binder, A.; Montavon, G.; Lapuschkin, S.; and Müller, K.-R. 2016. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems* 28:2660–2673.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR. abs/:1312.6034*.

Socher, R.; Huval, B.; Manning, C. D.; and Ng, A. Y. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*, 1201–1211.

Wang, L.; Cao, Z.; de Melo, G.; and Liu, Z. 2016. Relation classification via multi-level attention cnns. In *ACL*, 1298–1307.

Xu, K.; Feng, Y.; Huang, S.; and Zhao, D. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. In *EMNLP*, 536–540.

Yu, M.; Gormley, M.; and Dredze, M. 2014. Factor-based compositional embedding models. In *NIPS*, 95–101.

Zaidan, O.; Eisner, J.; and Piatko, C. D. 2007. Using annotator rationales to improve machine learning for text categorization. In *NAACL HLT*, 260–267.

Zeng, D.; Liu, K.; Lai, S.; Zhou, G.; and Zhao, J. 2014. Relation classification via convolutional deep neural network. In *COLING*, 2335–2344.

Zhang, Y.; Marshall, I.; and Wallace, B. C. 2016. Rationale-augmented convolutional neural networks for text classification. In *EMNLP*, 795–804.