

# Weighted Automata Extraction and Explanation of Recurrent Neural Networks for Natural Language Tasks

Zeming Wei<sup>a</sup>, Xiyue Zhang<sup>b</sup>, Yihao Zhang<sup>a</sup>, Meng Sun<sup>a,\*</sup>

<sup>a</sup>*School of Mathematical Sciences, Peking University, Beijing, China*

<sup>b</sup>*Department of Computer Science, University of Oxford, Oxford, United Kingdom*

---

## Abstract

Recurrent Neural Networks (RNNs) have achieved tremendous success in processing sequential data, yet understanding and analyzing their behaviours remains a significant challenge. To this end, many efforts have been made to extract finite automata from RNNs, which are more amenable for analysis and explanation. However, existing approaches like exact learning and compositional approaches for model extraction have limitations in either scalability or precision. In this paper, we propose a novel framework of Weighted Finite Automata (WFA) extraction and explanation to tackle the limitations for natural language tasks. First, to address the transition sparsity and context loss problems we identified in WFA extraction for natural language tasks, we propose an empirical method to complement missing rules in the transition diagram, and adjust transition matrices to enhance the context-awareness of the WFA. We also propose two data augmentation tactics to track more dynamic behaviours of RNN, which further allows us to improve the extraction precision. Based on the extracted model, we propose an explanation method for RNNs including a word embedding method – Transition Matrix Embeddings (TME) and TME-based task oriented explanation for the target RNN. Our evaluation demonstrates the advantage of our method in extraction precision than existing approaches, and the effectiveness of TME-based explanation method in applications to pretraining and adversarial example generation.

*Keywords:* Abstraction, Explanation, Weighted Finite Automata, Natural Languages, Recurrent Neural Networks

---

\*Corresponding author: sunm@pku.edu.cn

## 1. Introduction

In the last decade, deep learning (DL) has been widely deployed in a range of applications, such as image processing [1], speech recognition [2] and natural language processing [3]. In particular, recurrent neural networks (RNNs) achieve great success in sequential data processing, e.g., time series forecasting [4], text classification [5] and language translation [6]. However, the complex internal design and gate control of RNNs make the interpretation and analysis of their behaviours rather challenging.

Recently, much progress has been made to abstract RNN as a finite automaton, that is, a finite-state model with explicit states and transition matrix to characterize the behaviours of RNN in processing sequential data. Up to the present, a series of extraction approaches leverage explicit learning algorithms (e.g.,  $L^*$  algorithm [7]) to extract a surrogate model of RNN. Such exact learning procedure has achieved great success in capturing the state dynamics of RNNs when processing formal languages [8; 9; 10]. However, the computational complexity of the exact learning algorithm limits its scalability to construct abstract models from RNNs for natural language tasks. Another technical line for automata extraction from RNNs is the compositional approach, which uses unsupervised learning algorithms to obtain discrete partitions of RNNs' state vectors and construct the transition diagram based on the concrete state dynamics of RNNs. This approach demonstrates better scalability and has been applied to robustness analysis and repairment of RNNs on large-scale tasks [11; 12; 13; 14; 15; 16], but falls short in extraction precision. A precise and scalable extraction approach for RNNs in the context of natural language tasks is needed.

Regarding model-based explanation, current extraction methods are limited to utilizing finite automata as a global interpretable model with explicit states and transition rules for RNNs. The information extracted in the transition diagram of automata is not fully exploited in understanding RNN behaviors for natural language tasks. In particular, given that the alphabet size of natural language datasets is quite large, the extracted rules in the transition matrix are difficult to grasp and interpret. A more comprehensible explanation method that can effectively exploits the extracted information to assist in understanding RNN behaviors remains underexplored.

In this paper, we propose a general framework of Weighted Finite Automata (WFA) extraction and explanation for RNNs to tackle the above challenges. To address the first challenge, we propose a complete pipeline to extract more precise automata for RNNs in the context of natural language tasks. We identify two problems that cause precision deficiency in natural language tasks: (1) *transition sparsity*: the transition dynamics are usually

sparse in natural language tasks, due to the large alphabet size and the dependency on a finite set of (sequential) data in the extraction procedure. (2) *context loss*: the tracking of long-term context of RNNs (e.g., LSTM networks [17]) is inevitably compromised due to the abstraction. To deal with the transition sparsity problem, we propose a method to fill in the missing transition rules based on the semantics of abstract states. We also propose two tactics to augment the data samples to learn more transition behaviours of RNNs, which further alleviates the transition sparsity problem. To enhance the context awareness of WFAs, we adjust the transition matrices to preserve partial context information from the previous states.

To address the second challenge, we utilize the extracted WFAs to interpret the behaviours of RNNs. Motivated by the observation that the transition matrices of the extracted WFAs capture the behaviour of the source RNNs, we propose a word embedding method – Transition Matrix Embeddings (TME) to construct task-oriented explanations for the target RNNs. Further, by leveraging the information captured in TME, we propose a global explanation method for word attribution to RNNs’ decisions and a contrastive method to investigate the difference between task-oriented TME and pretrained word embeddings (e.g., Glove [18]). We validate the effectiveness of the contrastive explanation with applications to pretraining boost and adversarial example generation<sup>1</sup>.

We summarize our contributions as follows:

- (a) We propose a complete WFA extraction algorithm from RNNs designed for natural language tasks.
- (b) Experiments on benchmark datasets demonstrate that the proposed heuristic methods effectively improve the extraction precision by alleviating the transition sparsity and context loss problems.
- (c) We propose a novel word embedding – Transition Matrix Embeddings (TME), based on which a global explanation method for word attribution and a contrastive approach for task-oriented explanation of RNNs are proposed.

The organization of this paper is as follows. In Section 2, we present preliminaries about recurrent neural networks, weighted finite automata, and related notations and concepts. In Section 3, we present our transition rule extraction approach, including an overview on the automata extraction procedure, the transition rule complement method for transition sparsity, the

---

<sup>1</sup>Code is available at [https://github.com/weizeming/Extract\\_WFA\\_from\\_RNN\\_for\\_NL](https://github.com/weizeming/Extract_WFA_from_RNN_for_NL)

transition rule adjustment method for context-awareness enhancement, and the data augmentation tactics. In Section 4, we present the experimental evaluation towards the extraction consistency of our approach on two natural language tasks. We introduce the transition matrix embedding based explanation framework for RNNs in Section 5, and discuss our extraction algorithm including computational complexity analysis and applicability to other RNNs at the end of Section 5. Finally, we discuss related works in Section 6 and conclude our work in Section 7.

## 2. Preliminaries

In this section, we present the notations and definitions that will be used throughout the paper. Given a finite alphabet  $\Sigma$ , we use  $\Sigma^*$  to denote the set of sequences over  $\Sigma$  and  $\varepsilon$  to denote the empty sequence. For  $w \in \Sigma^*$ , we use  $|w|$  to denote its length, its  $i$ -th word as  $w_i$  and its prefix with length  $i$  as  $w[:i]$ . For  $x \in \Sigma$ ,  $w \cdot x$  represents the concatenation of  $w$  and  $x$ .

**Definition 1** (RNN). *A Recurrent Neural Network (RNN) for natural languages is a tuple  $\mathcal{R} = (\mathcal{X}, \mathcal{S}, \mathcal{O}, f, p)$ , where  $\mathcal{X}$  is the input space;  $\mathcal{S}$  is the internal state space;  $\mathcal{O}$  is the probabilistic output space;  $f : \mathcal{S} \times \mathcal{X} \rightarrow \mathcal{S}$  is the transition function;  $p : \mathcal{S} \rightarrow \mathcal{O}$  is the prediction function.*

*RNN Configuration.* In this paper, we consider RNN as a black-box model and focus on its stepwise probabilistic output for each input sequence. The following definition of configuration characterizes the probabilistic outputs in response to a sequential input fed to RNN. Given an alphabet  $\Sigma$ , let  $\xi : \Sigma \rightarrow \mathcal{X}$  be the function that maps each word in  $\Sigma$  to its embedding vector in  $\mathcal{X}$ . We define  $f^* : \mathcal{S} \times \Sigma^* \rightarrow \mathcal{S}$  recursively as  $f^*(s_0, \xi(w \cdot x)) = f(f^*(s_0, \xi(w)), \xi(x))$  and  $f^*(s_0, \varepsilon) = s_0$ , where  $s_0$  is the initial state of  $\mathcal{R}$ . The RNN configuration  $\delta : \Sigma^* \rightarrow \mathcal{O}$  is defined as  $\delta(w) = p(f^*(s_0, w))$ .

*Output Trace.* To record the stepwise behavior of RNN when processing an input sequence  $w$ , we define the *Output Trace* of  $w$ , i.e., the probabilistic output sequence, as  $T(w) = \{\delta(w[:i])\}_{i=1}^{|w|}$ . The  $i$ -th item of  $T(w)$  indicates the probabilistic output given by  $\mathcal{R}$  after taking the prefix of  $w$  with length  $i$  as input.

**Definition 2** (WFA). *Given a finite alphabet  $\Sigma$ , a Weighted Finite Automaton (WFA) over  $\Sigma$  is a tuple  $\mathcal{A} = (\hat{S}, \Sigma, E, \hat{s}_0, I, F)$ , where  $\hat{S}$  is the finite set of abstract states;  $E = \{E_\sigma | \sigma \in \Sigma\}$  is the set of transition matrix  $E_\sigma$  with size  $|\hat{S}| \times |\hat{S}|$  for each token  $\sigma \in \Sigma$ ;  $\hat{s}_0 \in \hat{S}$  is the initial state;  $I$  is the initial vector, a row vector with size  $|\hat{S}|$ ;  $F$  is the final vector, a column vector with size  $|\hat{S}|$ .*

*Abstract States.* Given a RNN  $\mathcal{R}$  and a dataset  $\mathcal{D}$ , let  $\hat{\mathcal{O}}$  denote all step-wise probabilistic outputs given by executing  $\mathcal{R}$  on  $\mathcal{D}$ , i.e.  $\hat{\mathcal{O}} = \bigcup_{w \in \mathcal{D}} T(w)$ .

The abstraction function  $\lambda : \hat{\mathcal{O}} \rightarrow \hat{\mathcal{S}}$  maps each probabilistic output to an abstract state  $\hat{s} \in \hat{\mathcal{S}}$ . As a result, the output set is divided into a number of abstract states by  $\lambda$ . For each  $\hat{s} \in \hat{\mathcal{S}}$ , the state  $\hat{s}$  has explicit semantics that the probabilistic outputs corresponding to  $\hat{s}$  has similar distribution. In this paper, we leverage the *k-means* algorithm to construct the abstraction function. We cluster all probabilistic outputs in  $\hat{\mathcal{O}}$  into some abstract states. In this way, we construct the set of abstract states  $\hat{\mathcal{S}}$  with these discrete clusters and an initial state  $\hat{s}_0$ .

For a state  $\hat{s} \in \hat{\mathcal{S}}$ , we define the *center* of  $\hat{s}$  as the average value of the probabilistic outputs  $\hat{o} \in \hat{\mathcal{O}}$  which are mapped to  $\hat{s}$ . More formally, the center of  $\hat{s}$  is defined as follows:

$$\rho(\hat{s}) = \text{Avg}_{\lambda(\hat{o})=\hat{s}} \{\hat{o}\}.$$

The center  $\rho(\hat{s})$  represents an approximation of the distribution tendency of probabilistic outputs  $\hat{o}$  in  $\hat{s}$ . We then use the center  $\rho(\hat{s})$  as its weight for each state  $\hat{s} \in \hat{\mathcal{S}}$ . The final vector  $F$  is thus formulated as  $(\rho(\hat{s}_0), \rho(\hat{s}_1), \dots, \rho(\hat{s}_{|\hat{\mathcal{S}}|-1}))^t$ .

*Abstract Transitions.* In order to capture the dynamic behavior of RNN  $\mathcal{R}$ , we define the abstract transition as a triple  $(\hat{s}, \sigma, \hat{s}')$  where the original state  $\hat{s}$  is the abstract state corresponding to a specific output  $y$ , i.e.  $\hat{s} = \lambda(y)$ ;  $\sigma$  is the next word of the input sequence to consume;  $\hat{s}'$  is the destination state  $\lambda(y')$  after  $\mathcal{R}$  reads  $\sigma$  and outputs  $y'$ . We use  $\mathcal{T}$  to denote the set of all abstract transitions tracked from the execution of  $\mathcal{R}$  on training samples.

*Abstract Transition Count Matrices.* For each word  $\sigma \in \Sigma$ , the abstract transition count matrix of  $\sigma$  is a matrix  $\hat{T}_\sigma$  with size  $|\hat{\mathcal{S}}| \times |\hat{\mathcal{S}}|$ . The count matrices record the number of times that each abstract transition is triggered. Given the set of abstract transitions  $\mathcal{T}$ , the count matrix of  $\sigma$  can be calculated as

$$\hat{T}_\sigma[i, j] = \mathcal{T}.\text{count}((\hat{s}_i, \sigma, \hat{s}_j)), \quad 1 \leq i, j \leq |\hat{\mathcal{S}}|.$$

As for the remaining components, the alphabet  $\Sigma$  is consistent with the alphabet of training set  $\mathcal{D}$ . The initial vector  $I$  is formulated according to the initial state  $\hat{s}_0$ .

For an input sequence  $w = w_1 w_2 \dots w_n \in \Sigma^*$ , the WFA will calculate its weight following

$$I \cdot E_{w_1} \cdot E_{w_2} \dots E_{w_n} \cdot F.$$

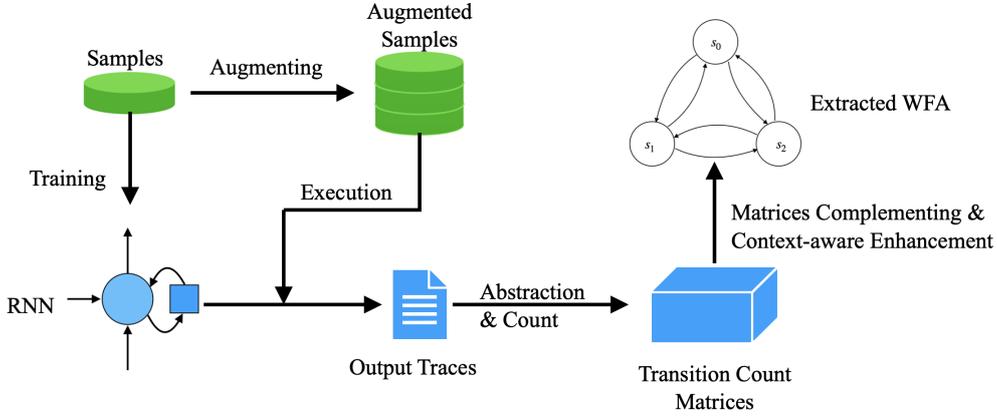


Figure 1: An illustration of our approach to extracting WFA from RNN.

### 3. Weighted Automata Extraction Scheme

#### 3.1. Overview

We present the workflow of our extraction procedure in Figure 1. As the first step, we generate augmented sample set  $\mathcal{D}$  from the original training set  $\mathcal{D}_0$  to enrich the transition dynamics of RNN behaviours and alleviate the transition sparsity. Then, we execute RNN  $\mathcal{R}$  on the augmented sample set  $\mathcal{D}$ , and record the probabilistic output trace  $T(w)$  of each input sentence  $w \in \mathcal{D}$ . With the output set  $\hat{O} = \bigcup_{w \in \mathcal{D}} T(w)$ , we cluster the probabilistic outputs into abstract states  $\hat{S}$ , and generate abstract transitions  $\mathcal{T}$  from the output traces  $\{T(w) | w \in \mathcal{D}\}$ . All transitions constitute the abstract transition count matrices  $\hat{T}_\sigma$  for all  $\sigma \in \Sigma$ .

Next, we construct the transition matrices  $E = \{E_\sigma | \sigma \in \Sigma\}$ . Based on the abstract states  $\hat{S}$  and count matrices  $\hat{T}$ , we construct the transition matrix  $E_\sigma$  for each word  $\sigma \in \Sigma$ . Specifically, we use frequencies to calculate the transition probabilities. Suppose that there are  $n$  abstract states in  $\hat{S}$ . The  $i$ -th row of  $E_\sigma$ , which indicates the probabilistic transition distribution over states when  $\mathcal{R}$  is in state  $\hat{s}_i$  and consumes  $\sigma$ , is calculated as

$$E_\sigma[i, j] = \frac{\hat{T}_\sigma[i, j]}{\sum_{k=1}^n \hat{T}_\sigma[i, k]}. \quad (1)$$

This empirical rule faces the problem that the denominator of (1) could be zero, which means that the word  $\sigma$  never appears when the RNN  $\mathcal{R}$  is in abstract state  $\hat{s}_i$ . In this case, one should decide how to fill in the transition

rule of the *missing rows* in  $E_\sigma$ . In Section 3.2, we present a novel approach for transition rule complement. Further, to preserve more contextual information during processing the input sequence, we propose an approach to enhancing the context-awareness of WFA by adjusting the transition matrices, which is presented in Section 3.3.

### 3.2. Missing Rows Complement

Existing approaches for transition rule extraction usually face the problem of transition sparsity, i.e., *missing rows* in the transition diagram. In the context of formal languages, the probability of the occurrence of missing rows is quite low, since the size of the alphabet is small and each token in the alphabet can appear sufficient number of times. However, in the context of natural language processing, the occurrence of missing rows is quite frequent. The following proposition gives an approximation of the occurrence frequency of missing rows.

**Proposition 1.** *Assume an alphabet  $\Sigma$  with  $m = |\Sigma|$  words, a natural language dataset  $\mathcal{D}$  over  $\Sigma$  which has  $N$  words in total, a RNN  $\mathcal{R}$  trained on  $\mathcal{D}$ , the extracted abstract states  $\hat{S}$  and transitions  $\mathcal{T}$ . Let  $\sigma_i$  denote the  $i$ -th most frequent word occurred in  $\mathcal{D}$  and  $t_i = \mathcal{T}.\text{count}((*, \sigma_i, *))$  indicates the occurrence times of  $\sigma_i$  in  $\mathcal{D}$ . The median of  $\{t_i | 1 \leq i \leq m\}$  can be estimated as*

$$t_{\lfloor \frac{m}{2} \rfloor} = \frac{2N}{m \cdot \ln m}.$$

*Proof.* The Zipf’s law [19] shows that

$$\frac{t_i}{N} \approx \frac{i^{-1}}{\sum_{k=1}^m k^{-1}}.$$

Note that  $\sum_{k=1}^m k^{-1} \approx \ln m$  and take  $i$  to be  $\frac{m}{2}$ , we complete our proof. □

**Example 1.** *In the QC news dataset [20], which has  $m = 20317$  words in its alphabet and  $N = 205927$  words in total, the median of  $\{t_i\}$  is approximated to  $\frac{2N}{m \cdot \ln m} \approx 2$ . This indicates that about half of  $E_\sigma$  are constructed with no more than 2 transitions. In practice, the number of abstract states is usually far more than the transition numbers of these words, making most rows of their transition matrices missing rows.*

Filling the missing row with  $\vec{0}$  is a simple solution, since no information were provided from the transitions. However, as estimated above, this solution will lead to the problem of transition sparsity, i.e., the transition matrices for uncommon words are nearly null. Consequently, if the input sequence includes some uncommon words, the weights over states tend to vanish. We refer to this solution as *null filling*.

Another simple idea is to use the uniform distribution over states for fairness. In [9], the uniform distribution is used as the transition distribution for unseen tokens in the context of formal language tasks. However, for natural language processing, this solution still loses information of the current word, despite that it avoids the weight vanishment over states. We refer to this solution as *uniform filling*. [21] uses the *synonym* transition distribution for an unseen token at a certain state. However, it increases the computation overhead when performing inference on test data, since it requires to calculate and sort the distance between the available tokens at a certain state and the unseen token.

To this end, we propose a novel approach to constructing the transition matrices based on two empirical observations. First, each abstract state  $\hat{s} \in \hat{S}$  has explicit semantics, i.e. the probabilistic distribution over labels, and similar abstract states tend to share more similar transition behaviours. The semantic distance between abstract states is defined as follows.

**Definition 3** (State Distance). *For two abstract states  $\hat{s}_1$  and  $\hat{s}_2$ , the distance between  $\hat{s}_1$  and  $\hat{s}_2$  is defined by the Euclidean distance between their center:*

$$\text{dist}(\hat{s}_1, \hat{s}_2) = \|\rho(\hat{s}_1) - \rho(\hat{s}_2)\|_2.$$

We calculate the distance between each pair of abstract states, which forms a *distance matrix*  $M$  where each element  $M[i, j] = \text{dist}(\hat{s}_i, \hat{s}_j)$  for  $1 \leq i, j \leq |\hat{S}|$ . For a missing row in  $E_\sigma$ , following the heuristics that similar abstract states are more likely to have similar behaviours, we observe the transition behaviours from other abstract states, and simulate the missing transition behaviours weighted by distance between states. Particularly, in order to avoid numerical underflow, we leverage *softmax* on distance to bias the weight to states that share more similarity. Formally, for a missing row  $E_\sigma[i]$ , the weight of information set for another row  $E_\sigma[j]$  is defined by  $e^{-M[i, j]}$ .

Second, it is also observed that sometimes the RNN just remains in the current state after reading a certain word. Intuitively, this is because part of words in the sentence do not deliver significant information in the task. Therefore, we consider simulating behaviours from other states whilst remaining in the current state with a certain probability.

In order to balance the trade-off between referring to behaviours from other states and remaining still, we introduce a hyper-parameter  $\beta$  named *reference rate*, such that when WFA is faced with a missing row, it has a probability of  $\beta$  to refer to the transition behaviours from other states, and in the meanwhile has a probability of  $1 - \beta$  to keep still. We select the parameter  $\beta$  according to the proportion of self-transitions, i.e., transitions  $(\hat{s}, \sigma, \hat{s}')$  in  $\mathcal{T}$  where  $\hat{s} = \hat{s}'$ .

To sum up, the complete transition rule for the missing row is

$$E_\sigma[i, j] = \beta \cdot \frac{\sum_{k=1}^n e^{-M[i,k]} \cdot \hat{T}_\sigma[k, j]}{\sum_{l=1}^n \sum_{k=1}^n e^{-M[i,k]} \cdot \hat{T}_\sigma[k, l]} + (1 - \beta) \cdot \delta_{i,j}. \quad (2)$$

Here  $\delta_{i,j}$  is the Kronecker symbol:

$$\delta_{i,j} = \begin{cases} 1, & j = i \\ 0, & j \neq i \end{cases}.$$

In practice, we can calculate  $\sum_{k=1}^n e^{-M[i,k]} \cdot \hat{T}_\sigma[k, j]$  for each  $j$  and then make division on their summation once and for all, which can reduce the computation overhead on transition rule extraction.

### 3.3. Context-Awareness Enhancement

For NLP tasks, the memorization of long-term context information is crucial. One of the advantages of RNN and its advanced design LSTM networks is the ability to capture long-term dependency. We expect the extracted WFA to simulate the step-wise behaviours of RNNs whilst keeping track of context information along with the state transition. To this end, we propose an approach to adjusting the transition matrix such that the WFA can remain in the current state with a certain probability.

Specifically, we select a hyper-parameter  $\alpha \in [0, 1]$  as the *static probability*. For each word  $\sigma \in \Sigma$  and its transition matrix  $E_\sigma$ , we replace the matrix with the *context-awareness enhanced matrix*  $\hat{E}_\sigma$  as follows:

$$\hat{E}_\sigma = \alpha \cdot I_n + (1 - \alpha) \cdot E_\sigma \quad (3)$$

where  $I_n$  is the identity matrix.

The context-awareness enhanced matrix has explicit semantics. When the WFA is in state  $\hat{s}_i$  and ready to process a new word  $\sigma$ , it has a probability

of  $\alpha$  (the *static probability*) to remain in  $\hat{s}_i$ , or follows the original transition distribution  $E_\sigma[i, j]$  with a probability  $1 - \alpha$ .

Here we present an illustration of how context-awareness enhanced matrices deliver long-term context information. Suppose that a context-awareness enhanced WFA  $\mathcal{A}$  is processing a sentence  $w \in \Sigma^*$  with length  $|w|$ . We denote  $d_i$  as the distribution over all abstract states after  $\mathcal{A}$  reads the prefix  $w[: i]$ , and particularly  $d_0 = I$  is the initial vector of  $\mathcal{A}$ . We use  $Z_i$  to denote the decision made by  $\mathcal{A}$  based on  $d_{i-1}$  and the original transition matrix  $E_{w_i}$ . Formally,  $d_i = d_{i-1} \cdot \hat{E}_{w_i}$  and  $Z_i = d_{i-1} \cdot E_{w_i}$ .

The  $d_i$  can be regarded as the information obtained from the prefix  $w[: i]$  by  $\mathcal{A}$  before it consumes  $w_{i+1}$ , and  $Z_i$  can be considered as the decision made by  $\mathcal{A}$  after it reads  $w_i$ .

**Proposition 2.** *The  $i$ -th step-wise information  $d_i$  delivered by processing  $w[: i]$  contains the decision information  $Z_j$  of prefix  $w[: j]$  with a proportion of  $(1 - \alpha) \cdot \alpha^{i-j}$ ,  $1 \leq j \leq i$ .*

*Proof.* Since  $\hat{E}_{w_i} = \alpha \cdot I_n + (1 - \alpha) \cdot E_{w_i}$ , we can calculate that

$$d_i = d_{i-1} \cdot \hat{E}_{w_i} = d_{i-1} \cdot [\alpha \cdot I_n + (1 - \alpha) \cdot E_{w_i}] = \alpha \cdot d_{i-1} + (1 - \alpha) \cdot Z_i. \quad (4)$$

Using (4) recursively, we have

$$d_i = (1 - \alpha) \sum_{k=1}^i \alpha^{i-k} \cdot Z_k + \alpha^i \cdot I.$$

□

This shows the information delivered by  $w[: i]$  refers to the decision made by  $\mathcal{A}$  on each prefix included in  $w[: i]$ , and the portion vanishes exponentially. The effectiveness of the context-awareness enhancement method for transition matrix adjustment will be discussed in Section 4.

The following example presents the complete approach for transition rule extraction, i.e., to generate the transition matrix  $\hat{E}_\sigma$  with the missing row filled in and context enhanced, from the count matrix  $\hat{T}_\sigma$  for a word  $\sigma \in \Sigma$ .

**Example 2.** *Assume that there are three abstract states in  $\hat{S} = \{\hat{s}_1, \hat{s}_2, \hat{s}_3\}$ . Suppose the count matrix for  $\sigma$  is  $\hat{T}_\sigma$ .*

$$\hat{T}_\sigma = \begin{bmatrix} 1 & 3 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, E_\sigma = \begin{bmatrix} 0.25 & 0.75 & 0 \\ 0.5 & 0.5 & 0 \\ 0.15 & 0.35 & 0.5 \end{bmatrix}, \hat{E}_\sigma = \begin{bmatrix} 0.4 & 0.6 & 0 \\ 0.4 & 0.6 & 0 \\ 0.12 & 0.28 & 0.6 \end{bmatrix}.$$

For the first two rows (states), there exist transitions for  $\sigma$ , thus we can calculate the transition distribution of these two rows in  $E_\sigma$  in the usual way. However, the third row is a missing row. We set the reference rate as  $\beta = 0.5$ , and suppose that the distance between states satisfies  $e^{-M[1,3]} = 2e^{-M[2,3]}$ , generally indicating the distance between  $\hat{s}_1$  and  $\hat{s}_3$  is nearer than  $\hat{s}_2$  and  $\hat{s}_3$ . With the transitions from  $\hat{s}_1$  and  $\hat{s}_2$ , we can complement the transition rule of the third row in  $E_\sigma$  through (2). The result shows that the behavior from  $\hat{s}_3$  is more similar to  $\hat{s}_1$  than  $\hat{s}_2$ , due to the smaller distance. Finally, we construct  $\hat{E}_\sigma$  with  $E_\sigma$ . Here we set the static probability  $\alpha = 0.2$ , thus  $\hat{E}_\sigma = 0.2 \cdot I_3 + 0.8 \cdot E_\sigma$ . The result shows that the WFA with  $\hat{E}_\sigma$  has higher probability to remain in the current state after consuming  $\sigma$ , which can preserve more information from the prefix before  $\sigma$ .

### 3.4. Data Augmentation

Our proposed approach for transition rule extraction provides a solution to the transition sparsity problem. Still, we hope to learn more dynamic transition behaviours from the target RNN, especially for the words with relatively low frequency to characterize their transition dynamics sufficiently based on the finite data samples. Different from formal languages, we can generate more natural language samples automatically, as long as the augmented sequential data are reasonable with clear semantics and compatible with the original learning task. Based on the augmented samples, we are able to track more behaviours of the RNN and build the abstract model with higher precision. In this section, we introduce two data augmentation tactics for natural language processing tasks: *Synonym Replacement* and *Dropout*.

*Synonym Replacement.* Based on the distance quantization among the word embedding vectors, we can obtain a list of synonyms for each word in  $\Sigma$ . For a word  $\sigma \in \Sigma$ , the *synonyms* of  $w$  are defined as the top- $k$  most similar words of  $\sigma$  in  $\Sigma$ , where  $k$  is a hyper-parameter and we set  $k$  to 5 by default based on an empirical observation that top-5 similar words are sufficiently reasonable to keep the semantics. The similarity among the words is calculated based on the Euclidean distance between the word embedding vectors over  $\Sigma$ .

Given a dataset  $\mathcal{D}_0$  over  $\Sigma$ , for each sentence  $w \in \mathcal{D}_0$ , we generate a new sentence  $w'$  by replacing some words in  $w$  with their synonyms. Specifically, each word is replaced by a randomly selected synonym in its top- $k$  synonyms with probability  $p_r$  (0.4 by default).

*Dropout.* Inspired by the regularization technique *dropout*, we also propose a similar tactic to generate new sentences from  $\mathcal{D}_0$ . Initially, we introduce a new word named *unknown word* and denote it as  $\langle \mathbf{unk} \rangle$ . For the sentence

$w \in \mathcal{D}_0$  that has been processed by synonym replacing, we further replace the words that haven't been replaced with  $\langle \mathbf{unk} \rangle$  with a certain probability  $p_d$  (0.2 by default). Finally, new sentences generated by both synonym replacement and dropout form the augmented dataset  $\mathcal{D}$ .

With the dropout tactic, we can observe the behaviours of RNNs when it processes an unknown word  $\hat{\sigma} \notin \Sigma$  that hasn't appeared in  $\mathcal{D}_0$ . Therefore, the extracted WFA can also have better generalization ability. The complete pipeline of the data augmentation algorithm is elaborated in Algorithm 1. Note that the *rand()* function samples from  $[0, 1]$  in a uniform manner.

---

**Algorithm 1:** Data Augmentation for Transition Rule Extraction

---

**Input** : Original dataset  $\mathcal{D}_0$ , hyper-parameter  $k = 5$ ,  $p_r = 0.4$ ,  
 $p_d = 0.2$

**Output:** Augmented dataset  $\mathcal{D}$

- 1 Obtain the synonyms  $\sigma_1, \sigma_2, \dots, \sigma_k$  of each word  $\sigma \in w$  in the vocabulary of  $\mathcal{D}$ ;
- 2  $\mathcal{D} \leftarrow \{\}$ ;
- 3 **for** each sentence  $w \in \mathcal{D}_0$  **do**
- 4     **for** each word  $\sigma \in w$  **do**
- 5         **if** *rand()*  $< p_r$  **then**
- 6             Replace  $\sigma$  with selected synonym from  $\{\sigma_1, \sigma_2, \dots, \sigma_k\}$ ;
- 7         **end**
- 8         **else**
- 9             **if** *rand()*  $< p_d$  **then**
- 10             Replace  $\sigma$  with  $\langle \mathbf{unk} \rangle$ ;
- 11             **end**
- 12         **end**
- 13     **end**
- 14     Obtain a new sentence  $w'$ , and add  $w'$  to  $\mathcal{D}$ ;
- 15 **end**
- 16 **return**  $\mathcal{D}$ ;

---

We illustrate the above data augmentation algorithm using the following example to generate a new sentence  $w'$  from  $\mathcal{D}_0$ .

**Example 3.** Consider a sentence  $w$  from the original training set  $\mathcal{D}_0$ ,  $w = [I, \text{'really'}, \text{'like'}, \text{'this'}, \text{'movie'}]$ . First, the word 'like' is chosen to be replaced by one of its synonym 'appreciate'. Next, the word 'really' is dropped from the sentence, i.e. replaced by the unknown word  $\langle \mathbf{unk} \rangle$ . Finally, we get a new sentence  $w' = [I, \langle \mathbf{unk} \rangle, \text{'appreciate'}, \text{'this'}, \text{'movie'}]$  and put it into the augmented dataset  $\mathcal{D}$ .

Since the word ‘appreciate’ may be an uncommon word in  $\Sigma$ , we can capture new transition information provided by RNNs. We can also capture the behavior of RNN when it reads an unknown word after the prefix [‘I’].

Note that the role of *data augmentation* in our extraction approach is different from that used in the training phase of RNNs. While data augmentation used in the training phase aims to improve the performance of RNNs, the goal of data augmentation in this work is to improve the WFA extraction precision. To this end, we use data augmentation in the testing phase to extract more transition dynamics to construct the abstract model.

## 4. Evaluation

In this section, we evaluate our extraction approach on two natural language datasets and demonstrate its performance on precision and scalability.

### 4.1. Datasets and RNNs

We select two popular datasets for NLP tasks and train the target RNNs on them.

1. The CogComp QC Dataset (abbrev. QC) [20] contains news titles which are labeled with different topics. The dataset is divided into a training set containing 20k samples and a test set containing 8k samples. Each sample is labeled with one of seven categories. We train an LSTM model  $\mathcal{R}$  on the training set, which achieves an accuracy of 81% on the test set.
2. The Jigsaw Toxic Comment Dataset (abbrev. Toxic) [22] contains comments from Wikipedia’s talk page edits, with each comment labeled toxic or not. We select 25k non-toxic samples and toxic samples respectively, and divide them into the training set and test set in a ratio of four to one. We train an LSTM model which achieves 90% accuracy on the test set.

*Metrics.* We use *Consistency Rate (CR)* and *Jensen–Shannon Divergence (JSD)* as our evaluation metrics. For a sentence in the test set  $w \in \mathcal{D}_{test}$ , we use  $\mathcal{R}(w)[i]$  and  $\mathcal{A}(w)[i]$  to denote the prediction score on class  $i$  of the RNNs and WFA, respectively. The *Consistency Rate* measures the consistency of the output decision between the two models, which is formally defined as

$$CR = \frac{|\{w \in \mathcal{D}_{test} : \arg \max_i \mathcal{A}(w)[i] = \arg \max_i \mathcal{R}(w)[i]\}|}{|\mathcal{D}_{test}|}. \quad (5)$$

Dataset	QC			Toxic		
Metric	CR( $\uparrow$ )	JSD( $\downarrow$ )	Time(s)	CR( $\uparrow$ )	JSD( $\downarrow$ )	Time(s)
$\mathcal{A}_0$	0.26	0.25	47	0.57	0.09	167
$\mathcal{A}_U$	0.60	0.21	56	0.86	0.06	180
$\mathcal{A}_E$	<b>0.80</b>	<b>0.10</b>	70	<b>0.91</b>	<b>0.02</b>	200

Table 1: Evaluation results of different filling approaches on missing rows.

The *Jensen–Shannon Divergence* [23] measures the distance of two probability distributions, i.e., the outputs of WFA and RNN, which is formally defined as

$$JSD = \frac{1}{2} \sum_i (\mathcal{A}(w)[i] \log(\frac{2\mathcal{A}(w)[i]}{\mathcal{A}(w)[i] + \mathcal{R}(w)[i]}) + \mathcal{R}(w)[i] \log(\frac{2\mathcal{R}(w)[i]}{\mathcal{A}(w)[i] + \mathcal{R}(w)[i]})). \quad (6)$$

Note that the Consistency Rate measures the consistency of the classification decision between the WFA and the RNN, while Jensen–Shannon Divergence evaluates the similarity of the output probability distributions between the two models. These two metrics evaluate the consistency between the abstract model and RNN to a different degree. In this paper we mainly focus on the consistency of predicted labels, hence we apply Consistency Rate as our major measurement.

#### 4.2. Missing Rows Complementing

As discussed in Section 3.2, we take two approaches as baselines, the *null filling* and the *uniform filling*. The extracted WFA with these two approaches are denoted as  $\mathcal{A}_0$  and  $\mathcal{A}_U$ , respectively. The WFA extracted by our *empirical filling* approach is denoted as  $\mathcal{A}_E$ .

Table 1 shows the evaluation results of three rule filling approaches. We conduct the comparison experiments on QC and Toxic datasets and select the cluster number for state abstraction as 40 and 20 for the QC and Toxic datasets, respectively.

The three rows labeled with the type of WFA show the evaluation results of different approaches. For the  $\mathcal{A}_0$  based on nul filling , the WFA returns the weight of most sentences in  $\mathcal{D}$  with  $\vec{0}$ , which fails to provide sufficient information for prediction. For the QC dataset, only a quarter of sentences in the test set are classified correctly. The second row shows that the performance of  $\mathcal{A}_U$  is better than  $\mathcal{A}_0$ . The last row presents the evaluation result of  $\mathcal{A}_E$ , which fills in the missing rows by our approach. In this experiment, the hyper-parameter *reference rate* is set as  $\beta = 0.3$ . We can see that our empirical approach achieves significantly better accuracy, which is 20% and 5%

Rule	Config.	QC			Toxic		
		CR( $\uparrow$ )	JSD( $\downarrow$ )	Time(s)	CR( $\uparrow$ )	JSD( $\downarrow$ )	Time(s)
$\mathcal{A}_U$	None	0.60	<b>0.21</b>	56	0.86	<b>0.06</b>	180
	Context	<b>0.71</b>	0.22	64	<b>0.89</b>	<b>0.06</b>	191
$\mathcal{A}_E$	None	0.80	<b>0.10</b>	70	0.91	<b>0.02</b>	200
	Context	<b>0.82</b>	0.13	78	<b>0.92</b>	0.03	211

Table 2: Evaluation results of with and without context-awareness enhancement.

higher than uniform filling on the two datasets, respectively. As for JSD, we can see that our empirical approach also outperforms the baselines notably over both QC and Toxic datasets.

The columns labeled *Time* show the execution time of the whole extraction workflow, from tracking transitions to evaluation on test set, but not include the training time of RNNs. We can see that the extraction overhead of our approach ( $\mathcal{A}_E$ ) is about the same as  $\mathcal{A}_U$  and  $\mathcal{A}_0$ .

#### 4.3. Context-Awareness Enhancement

In this experiment, we leverage the context-awareness enhanced matrices when constructing the WFA. We adopt the same configuration on cluster numbers  $n$  as the comparison experiments above, i.e.  $n = 40$  and  $n = 20$ . The experiment results are summarized in Table 2. The columns titled *Config.* indicate if the extracted WFA leverage context-awareness matrices. We also take the WFA with different filling approaches, the uniform filling and empirical filling, into comparison. Experiments on null filling is omitted due to limited precision.

For the QC dataset, we set the *static probability* as  $\alpha = 0.4$ . The consistency rate of WFA  $\mathcal{A}_U$  improves 11% with the context-awareness enhancement, and  $\mathcal{A}_E$  improves 2%. As for the Toxic dataset, we take  $\alpha = 0.2$  and the consistency rate of the two WFA improves 3% and 1% respectively. This shows that the WFA with context-awareness enhancement remains more context information from the prefixes of sentences, making it simulate RNNs’ classification decision better. However, the WFA equipped with context-awareness enhancement exhibit larger JSD, which is caused by the fact that context-awareness enhancement reduces the transition magnitude, since larger  $\alpha$  leads to higher probability on remaining in the current state. This reveals a trade-off between the abstraction precision evaluated by decision label consistency and prediction score consistency.

Still, the context-awareness enhancement processing costs little time, since we only calculate the adjusting formula (3) for each  $E_\sigma$  in  $E$ . The

Rule	Data	QC			Toxic		
		CR( $\uparrow$ )	JSD( $\downarrow$ )	Time(s)	CR( $\uparrow$ )	JSD( $\downarrow$ )	Time(s)
$\mathcal{A}_U$	$\mathcal{D}_0$	0.71	0.22	64	0.89	0.06	191
	$\mathcal{D}$	<b>0.76</b>	<b>0.18</b>	81	<b>0.91</b>	<b>0.05</b>	295
$\mathcal{A}_E$	$\mathcal{D}_0$	0.82	0.13	78	0.92	0.03	211
	$\mathcal{D}$	<b>0.84</b>	<b>0.12</b>	85	<b>0.94</b>	<b>0.02</b>	315

Table 3: Evaluation results of with and without data augmentation.

additional extra time consumption is 8s for the QC dataset and 11s for the Toxic dataset.

#### 4.4. Data Augmentation

Finally, we evaluate the WFA extracted with transition behaviours from augmented data. Note that the two experiments above are based on the primitive dataset  $\mathcal{D}_0$ . In this experiment, we leverage the data augmentation tactics to generate the augmented training set  $\mathcal{D}$ , and extract WFA with data samples from  $\mathcal{D}$ . In order to get best performance, we build WFA with context-awareness enhanced matrices.

Table 3 shows the results of consistency rate of WFA extracted with and without augmented data. The rows labeled  $\mathcal{D}_0$  show the results of WFA that are extracted with the primitive training set, and the result from the augmented data is shown in rows labeled  $\mathcal{D}$ . With more transition behaviours tracked, the WFA extracted with  $\mathcal{D}$  demonstrates better precision. Specifically, the WFA extracted with both empirical filling and context-awareness enhancement achieves a further 2% increase in consistency rate on the two datasets. In addition, the extractions with augmented data also exhibit better JSD.

To summarize, by using our transition rule extraction approach, the consistency rate of extracted WFA on the QC dataset and the Toxic dataset achieves 84% and 94%, respectively. Taking the primitive extraction algorithm with uniform filling as baseline, of which experimental results in terms of CR are 60% and 86%, our approach achieves an improvement of 22% and 8% in consistency rate. Regarding the Jensen–Shannon Divergence, though there is a little drop made by the context-awareness enhancement, our approach still outperforms the baseline methods significantly. Taking uniform filling for comparison, our overall approach improves the JSD from 0.21 to 0.12 on QC dataset and 0.06 to 0.02 on Toxic dataset. For the time complexity, the time consumption of our approach increases from 56s to 81s on QC dataset, and from 180s to 315s on Toxic dataset. There is no significant time cost increase when adopting our approach for complicated natural language

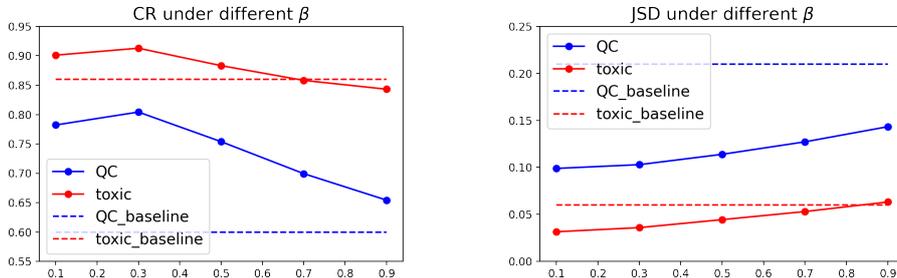


Figure 2: CR and JSD on the two datasets under different  $\beta$ .

tasks. We can conclude that our transition rule extraction approach makes better approximation of RNNs, and is also efficient enough to be applied to practical applications for large-scale natural language tasks.

#### 4.5. Parameter Effect Evaluation

In this section, we conduct experiments to evaluate the impact of the hyper-parameters on the validity of our extraction approach, including the reference rate  $\beta$ , static probability  $\alpha$ , and the number of cluster  $K$ .

*Reference rate  $\beta$ .* We first evaluate the impact of the *reference rate*. To this end, we set  $\beta$  to different values from  $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ . Meanwhile, we set  $\alpha$  to a fixed value 0. The results are shown in Figure 2, where we take the uniform filling as baseline (the dotted lines). We observe that our filling method outperforms uniform filling for a large range of parameter values (less than 0.7), under both CR and JSD metrics. A relatively small  $\beta$  (e.g., less than 0.5) leads to better extraction precision.

*Static probability  $\alpha$ .* Similarly, we conduct an experiment to evaluate the impact of different *static probability* values  $\alpha \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$  on the performance of our approach. We set  $\beta$  to 0.3 based on the above evaluation results. The results are illustrated in Figure 3. Compared with the case of  $\alpha = 0$  where we do not apply the context-awareness enhancement, it leads to improvements on the CRs when setting  $\alpha$  to values from  $\{0.2, 0.4\}$ . Meanwhile, as discussed before, context-awareness enhancement reduces the transition scale of WFA, which leads to performance degradation in terms of JSD. This reveals a trade-off between CR and JSD among different selections on  $\alpha$ . Based on the results, we suggest setting  $\alpha$  to a small positive value (less than 0.4).

*Cluster number  $K$ .* Finally, we evaluate the impact of cluster number  $K \in \{10, 20, 30, 40, 50\}$  on the performance of our approach. The results are shown

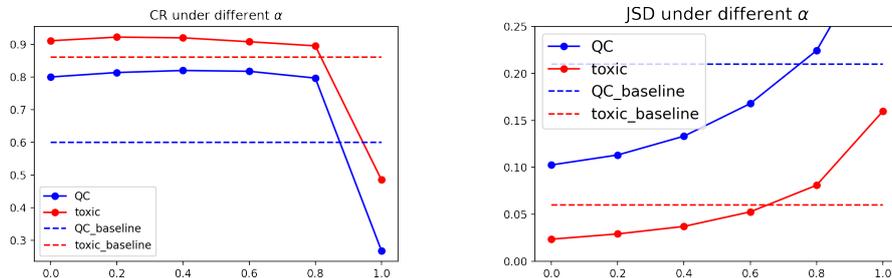


Figure 3: CR and JSD on the two datasets under different  $\alpha$ .

in Figure 4, where our approach is denoted by  $A_E$ , and the uniform filling and null filling are denoted as  $A_U, A_0$  respectively. We can see that our approach outperforms the baselines in all cases. Note that as  $K$  increases from 10 to 50, the performance of  $A_U$  and  $A_0$  consistently decreases, which is caused by the transition sparsity problem we have identified. The evaluation results demonstrate the robustness of our method against the cluster number  $K$ .

## 5. Weighted Automata-based Explanation of RNNs

In this section, we propose a novel explanation framework of RNNs for natural language tasks based on the extracted WFA. In the explanation framework, we consider using transition matrix<sup>2</sup>  $E_\sigma$  for each word  $\sigma$  as its word embedding, named as Transition Matrix Embeddings (TME). In the following, we first introduce TME and explain its difference from traditional pretrained word embeddings. Next, we present a global explanation and contrastive explanation method based on TME to interpret the behaviours of RNNs: (1) *global explanation*: we use TME to calculate the word-wise attribution of RNN’s decisions, (2) *contrastive explanation*: we compare TME with the conventional word embedding method to analyze the task-oriented word semantics learned by RNNs. We also reveal two intriguing properties of RNNs identified by the contrastive method, and validate the effectiveness of the proposed embedding and explanation framework for RNNs by applying it to pretraining and adversarial example generation.

### 5.1. Transition Matrix as Word Embeddings

Suppose the extracted WFA  $\mathcal{A}$  from RNN  $\mathcal{R}$  has  $n = |\hat{S}|$  words. For each word  $\sigma$  and its corresponding transition matrix  $E_\sigma \in \mathbb{R}^{n \times n}$  in  $\mathcal{A}$ , recall that

<sup>2</sup>In this section, we focus on the final extracted transition matrix  $\hat{E}_\sigma$ , and abuse the notation  $E_\sigma$  for the sake of simplicity.

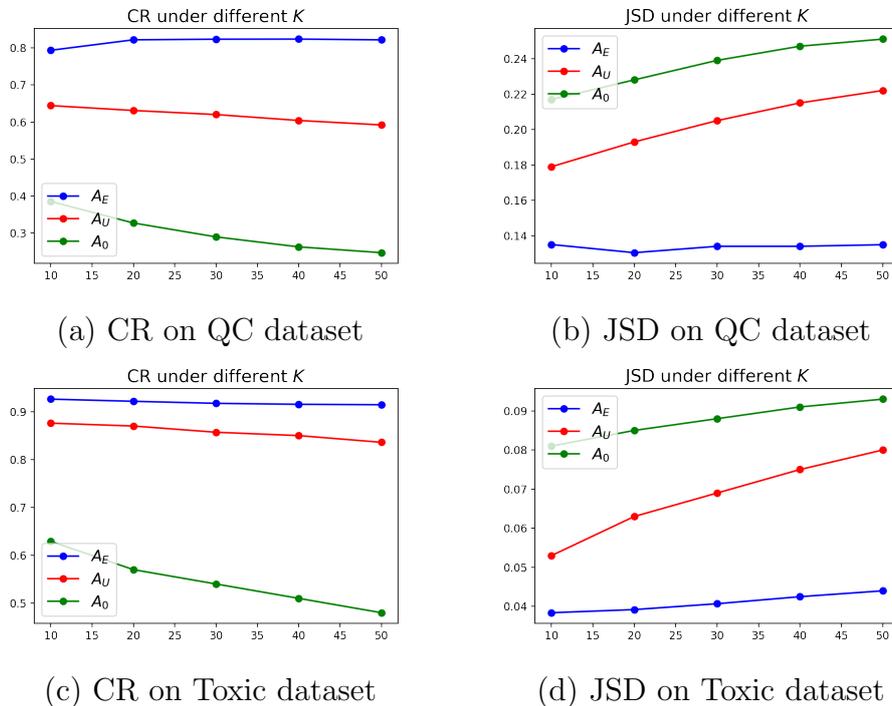


Figure 4: Overall comparison under different  $K$ .

the  $(i, j)$ -th element of  $E_\sigma$  represents the transition probability of  $\mathcal{A}$  from  $s_i$  to  $s_j$  after reading word  $\sigma$ , which is an approximation of the transition probability of  $\mathcal{R}$  between these two states. Therefore, if two words share similar transition matrices, they trigger similar behaviours of RNN  $\mathcal{R}$ , and hence represent similar semantics from the RNN  $\mathcal{R}$ 's perspective for the current task.

This observation motivates us using the transition matrices to craft word embeddings. In order to obtain the task-oriented embedding vector, we flatten the transition matrix  $E_\sigma \in \mathbb{R}^{n \times n}$  into a vector  $\mathbf{e}_\sigma$  of  $N = n^2$  dimension:

$$\mathbf{e}_\sigma[j + n \cdot (i - 1)] = E_\sigma[i, j], \quad \text{for } 1 \leq i, j \leq n, \quad (7)$$

which we refer to as Transition Matrix Embeddings (TME).

Note that TME are fundamentally different from the traditional pre-trained word embeddings, *e.g.* Word2vec [24], Gloves [18]. The TME characterize transition behaviours of RNNs when processing each word, which are task oriented (*e.g.*, text classification), while pre-trained embeddings are word-semantic oriented. Therefore, for two words  $\sigma_1$  and  $\sigma_2$ , they may share similar TME  $\mathbf{e}_{\sigma_1}$ ,  $\mathbf{e}_{\sigma_2}$ , yet represent different semantics and hence their embedding differences in pre-trained embeddings may be large. We detail such

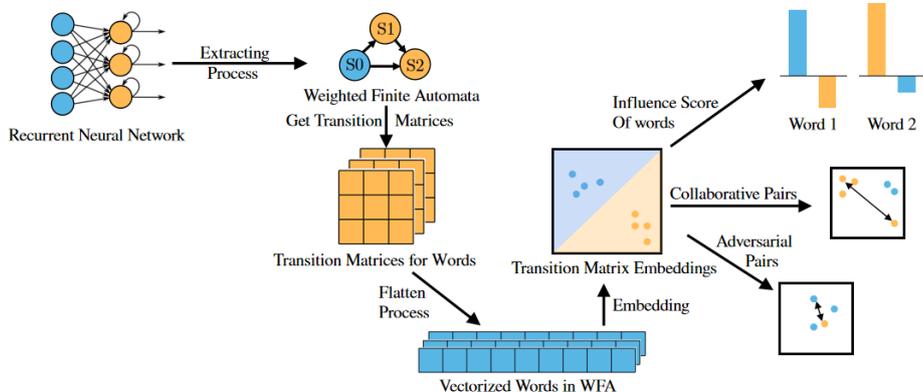


Figure 5: WFA-based explanation framework.

cases in Section 5.3. Further, the applications of TME are different from the general pretrained embeddings. The extracted TME can be seen as a global explanation of the source RNN  $\mathcal{R}$  (see Section 5.2 for details), which aids us for understanding the decision logic of the source RNN.

### 5.2. Word-Wise Attribution with TME

We introduce a global explanation method based on TME for analyzing the decision attribution of the source RNN  $\mathcal{R}$ . We investigate the impact of each word  $\sigma$  on the decision of  $\mathcal{R}$  based on its TME  $e_\sigma$ . Recall that each abstract state  $\hat{s}$  has an explicit semantic represented by its center  $\rho(\hat{s})$ , the probability distribution of labels. Therefore, each transition between two states, such as  $\hat{s}_1 \rightarrow \hat{s}_2$ , can be interpreted as a shift in the probability distribution of labels,  $\rho(\hat{s}_1) \rightarrow \rho(\hat{s}_2)$ . By multiplying the transition probability between states, which is saved in  $e_\sigma$ , we can calculate the average variation of prediction scores among labels after  $\mathcal{R}$  reading word  $\sigma$ . More formally, the variation contributed by the abstract transition  $(\hat{s}_i, \sigma, \hat{s}_j)$  is given by  $e_\sigma[j + n \cdot (i - 1)] \times (\rho(\hat{s}_j) - \rho(\hat{s}_i))$ .

Additionally, we take into account the uneven significance among all abstract states, where states that appear more frequently should be assigned larger weight. To reflect this, we calculate the frequency of each abstract state  $\hat{s}$  as  $u(\hat{s})$  and incorporate it into the computation of influence score as a weight. Formally, the *Influence Score* (IS) of word  $\sigma$  is formulated as

$$\text{IS}(\sigma) = \sum_{i=1}^n u(\hat{s}_i) \left\{ \sum_{j=1}^n e_\sigma[j + n \cdot (i - 1)] \times (\rho(\hat{s}_j) - \rho(\hat{s}_i)) \right\}. \quad (8)$$

For class  $i$ , the  $i$ -th element of influence score for word  $\sigma$  represents how this word impacts the decision of RNN on this class. Therefore, we can

Label	Category	Top-10 Influential Words
0	Sport	players, lockout, rangers, sox, knicks, basketball, coach, bruins, champions, djokovic
1	World	libyan, pakistan, yemen, sudan, gaddafi, syrian, egypt, mubarak, syria, afghans
2	US	florida, county, wildfire, layoffs, massachusetts, mid-atlantic, wildfires, cpsc, firefighters, blagojevich
3	Business	stocks, dollar, consumer, goldman, mortgage, wall, companies, rosneft, s&p, sec
4	Health	disease, crackers, cancer, asthma, patients, exercise, prevention, symptoms, therapy, obesity
5	Entertainment	idol, cannes, arthur, gotti, beaver, mcreery, mariah, lohan, baldwin, diana
6	Sci_tech	3ds, playstation, icloud, software, gmail, windows, tablets, linkedin, tablet, climate

Table 4: Top-10 Influential Words for 7 classes in the QC news Dataset [20]

investigate the influential words for the source RNN’s decisions by sorting the input words in descending order of IS. To demonstrate the effectiveness of the proposed influence analysis method, we compute the IS of all words in the vocabulary of the QC news dataset, and show the top-10 influential words for each class in Table 4. Due to the presence of inappropriate language in the Toxic dataset we exclude the experiment results on this dataset. From Table 4 we can see that for each category, its most influential words are highly correlated to that domain. This confirms that the proposed influence score based on TME can indeed identify the input features (words) that RNN  $\mathcal{R}$  relies on to make decisions on each class.

We can also use the TME to characterize the relative importance of an individual word for different labels. For instance, we show the influence scores of the words “Basketball”, “Dollar”, “Apple”, and “Happy” in Figure 6. As shown in Figure 6, the ISs of “Basketball” and “Dollar” demonstrate that they lead to high prediction tendency on class “Sport” and “Business”, respectively, which is strongly correlated to their semantic domains. In contrast, the word “Apple” shows high influence score on class “Business” and “Sci.tech”, which is consistent with the intuition that this word is active in both business and technology news. Finally, the word “Happy”, which has no particular influence on any class, demonstrates uniform IS on each class.

To sum up, the proposed TME provide a global explanation of the source RNN  $\mathcal{R}$ . As discussed above, by computing TME-based influential score, we can provide explanations of  $\mathcal{R}$  from both class-wise and word-wise perspec-

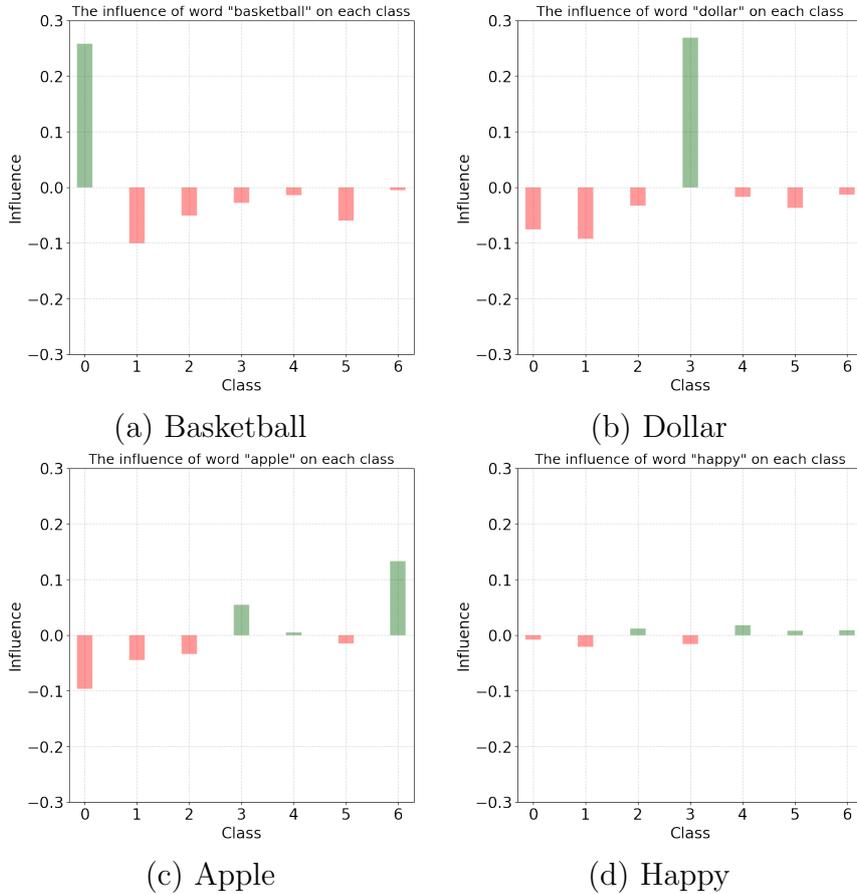


Figure 6: Influence Scores (ISs) visualization for some words.

tives.

### 5.3. Contrastive Word Relation

Based on the Transition Matrix Embeddings (TME), we propose a contrastive method to investigate the relations of words in TME and conventional word embeddings, which reveals two intriguing properties.

First, to demonstrate the difference between TME and Glove embeddings, we use t-SNE [25] to visualize the embedding vectors of TME and Glove, which is shown in Figure 7. Specifically, we select top-50 influential words from each class in QC news dataset with each color representing a class. We can see that the selected 350 ( $50 \times 7$ ) words demonstrate different clustering property in these two word embeddings. This shows our TME are quite different from pretrained word embeddings, wherein the word semantics are task-oriented.

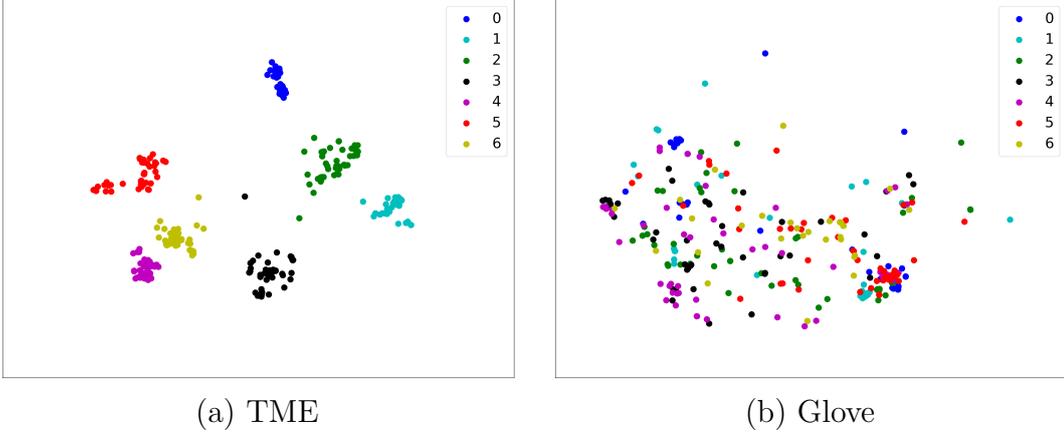


Figure 7: t-SNE visualization of two kinds of word embeddings on QC dataset. Each color represents a class.

We now characterize the contrastive relations between words in TME and the conventional word embeddings. We define  $\|\cdot\|$  as the  $p$ -norm of a matrix or a vector divided by the number of its elements, and we set  $p$  to 2. We compute the distance of two words given by TME and their conventional semantics, respectively. For words  $\sigma_1, \sigma_2$ , we define their transition distance as  $d_T(\sigma_1, \sigma_2) = \|\mathbf{e}_{\sigma_1} - \mathbf{e}_{\sigma_2}\|$ . In order to analyze their conventional semantic distance, we use the Glove [18] word embeddings  $\mathbf{g}_{\sigma_1}, \mathbf{g}_{\sigma_2}$ , and define the semantic distance as  $d_S(\sigma_1, \sigma_2) = \|\mathbf{g}_{\sigma_1} - \mathbf{g}_{\sigma_2}\|$ . By analyzing these two embedding distances between words, we find that there exist some contrastive word pairs demonstrating different properties. We formally define two types of contrastive word pairs in the following.

**Definition 4** ( $(\epsilon, \delta)$ -Collaborative Pair). *A  $(\epsilon, \delta)$ -Collaborative Pair is a pair of words  $(\sigma_1, \sigma_2)$  satisfying that*

$$d_T(\sigma_1, \sigma_2) \leq \epsilon, \quad d_S(\sigma_1, \sigma_2) \geq \delta. \quad (9)$$

Here  $\epsilon$  is a small positive number to guarantee that the word pair  $\sigma_1$  and  $\sigma_2$  trigger similar transition behaviours of the source RNN  $\mathcal{R}$ . On the other hand,  $\delta$  is a relatively larger positive number, indicating these two words have quite different semantics in terms of conventional embeddings. Hence the *collaborative pairs* are the word pairs that have distinct meanings, but are similar from the RNN  $\mathcal{R}$ 's perspective on the specific task. In contrast, the *adversarial pairs* are defined in a symmetry manner, which means the words share similar meanings, but are quite different from the RNN  $\mathcal{R}$ 's understanding in a particular task.

Label	Category	Collaborative Pairs	Adversarial Pairs
0	Sport	(‘lakers’, ‘wozniacki’)	(‘cup’, ‘cups’)
1	World	(‘yemen’, ‘gaddafi’)	(‘yemen’, ‘usa’)
2	US	(‘wildfire’, ‘texas’)	(‘wildfire’, ‘tsunami’)
3	Business	(‘wall’, ‘mortgage’)	(‘wall’, ‘behind’)
4	Health	(‘therapy’, ‘rice’)	(‘exercise’, ‘sports’)
5	Entertainment	(‘cannes’, ‘bieber’)	(‘diana’, ‘williams’)
6	Sci_tech	(‘climate’, ‘software’)	(‘windows’, ‘open’)

Table 5: Examples of Collaborative Pairs and Adversarial Pairs.

**Definition 5** ( $(\epsilon, \delta)$ -Adversarial Pair). An  $(\epsilon, \delta)$ -Adversarial Pair is a pair of words  $(\sigma_1, \sigma_2)$  satisfying that

$$d_T(\sigma_1, \sigma_2) \geq \delta, \quad d_S(\sigma_1, \sigma_2) \leq \epsilon. \quad (10)$$

The above contrastive pairs allow us to understand how RNN learns the semantics of the words. When a dataset and a task is given, the semantic of a word in the vocabulary is not learned fully obeying general word embedding, but *task-oriented*. To make the task-oriented word semantics clearer, we show some examples of  $(\epsilon, \delta)$ -Collaborative Pairs and Adversarial Pairs found by our algorithm in Table 5. The  $(\epsilon, \delta)$  is set to be  $(0.012, 0.1)$  for collaborative pairs, and  $(0.2, 0.01)$  for adversarial pairs. Note that the size of  $(\epsilon, \delta)$  for adversarial pairs is significantly different from that for collaborative pairs. In collaborative pairs,  $\epsilon$  is set to a relatively small positive value, ensuring that the embedding distances in RNN are small, while for adversarial pairs,  $\epsilon$  is set to a larger value to avoid strict constraints on semantic distance that would make the resulting adversarial words too similar. The value of  $\delta$  is also set for similar reasons. From these examples, we identify two intriguing properties of the source RNN. The *collaborative pairs* are the pair of words which the source RNN processes similarly with regard to the current task, but not synonyms in conventional semantics. On the other hand, the *adversarial pairs* are actually synonyms, but when considered in the current task, the behaviours of RNNs are triggered differently. These contrastive pairs capture the RNN’s specific understanding of word semantics, which are task-oriented.

Next, we analyze the adversarial pairs with a concrete example. Note that the collaborative pairs can be analyzed in a similar way. Consider the adversarial pair (“exercise”, “sports”) as an example, which are synonyms in general word semantics. But when we analyze their influence on RNN’s decisions, they demonstrate significant differences. The influence analysis results show that “exercise” is a word that has high influence score on class “Health”,

while “sports” is a word that is most influential to the “Sports” category. We further present an adversarial example generated by leveraging this adversarial pair. Consider the following sentence in the test set, “exercise helps her age swimmingly”, on which the RNN outputs “Health” with probability of 98.9%. When we feed the sentence “sports helps her age swimmingly” to the RNN instead, the output probability of category “Sport” raises up to 92.7%. However, the two sequences have nearly the same semantics. Based on the above result, we see that synonyms with regard to general embeddings are understood differently by RNNs. Therefore, TME and TME-based explanation can help us better understanding what the target RNN learns and how it makes decisions.

In this way, by identifying and analyzing the collaborative examples, we can understand what are task-oriented synonyms from the target RNN’s perspective, though they may be distinct in conventional embedding semantics. On the other hand, characterizing adversarial pairs provides explanations of the target RNN on distinguishing similar words in the context of the current task. We further validate the effectiveness of the contrastive pairs with the following two applications.

### 5.3.1. TME for RNN Pretraining

The identification of collaborative pairs reveals that TME is able to characterize task-oriented semantics, compared with the conventional embedding method like Glove. We next show the effectiveness of TME in boosting RNN training when serving as pretrained embeddings.

In the experiment, we consider training RNNs on the QC news dataset and Toxic dataset with three word embedding initialization strategies: (i) TME, (ii) Glove, and (iii) random initialization. Figure 8 shows the comparison results. We can see that the initialization with TME outperforms Glove and random initialization on convergence speed in terms of loss and accuracy on the test set, which validates the effectiveness of TME in boosting RNN pretraining.

Note that there is a step rise in accuracy observed during the training process. In fact, for general NLP tasks, neural networks tend to experience a rapidly initial improvement in accuracy and then reach a plateau as training progresses. In our benchmarks tasks, the initialization of word embedding vectors has a significant impact on the network’s ability to learn the correct patterns. It is only after the network learns the correct semantics from these embeddings, then the neural network can enter the phase of improvement in accuracy. As our pre-trained word embeddings are initialised with clearer semantics, the network is able to reach this phase of improvement at an earlier stage in the training process.

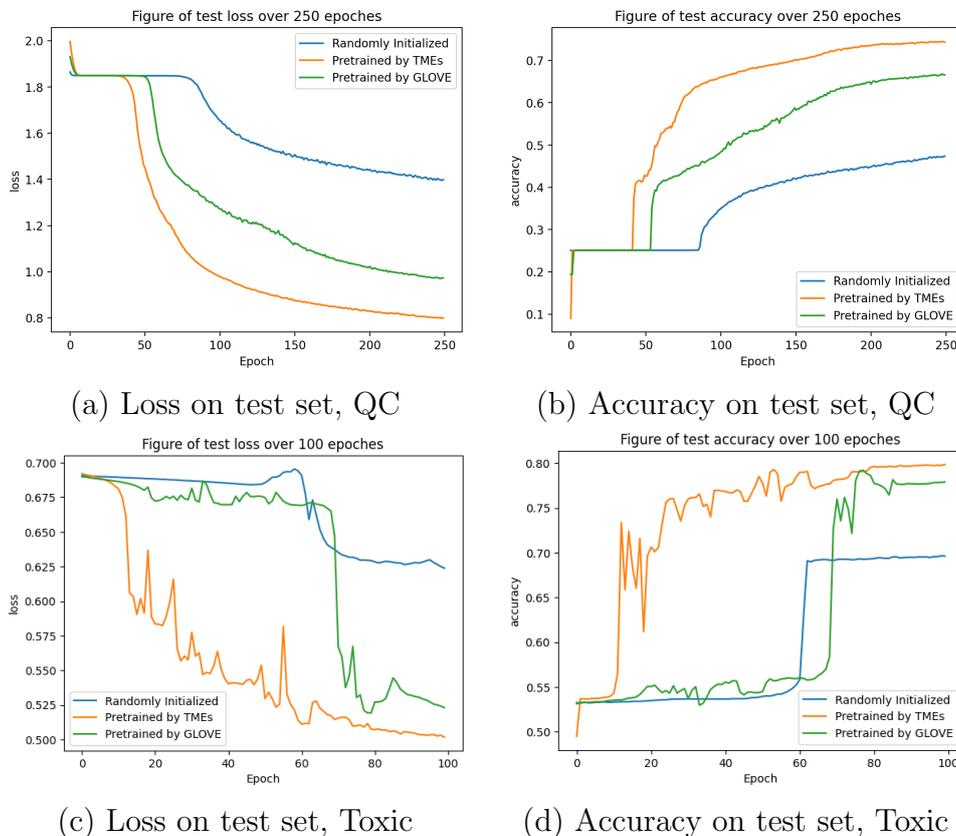


Figure 8: Comparison of three initialization strategies for RNN training.

### 5.3.2. TME for Adversarial Example Generation

Previous investigation has shown that TME can be utilized to identify adversarial pairs and decision vulnerabilities of RNNs. Inspired by the investigation results, we apply TME to generate adversarial examples for the source RNN. We perform comparison experiments of using TME and Glove as embeddings in crafting adversarial examples. To evaluate the effectiveness of different methods in adversarial example generation, we use *Attack Success Rate (ASR)* as the evaluation metric, namely the proportion of crafted sequences that successfully mislead the RNN to produce false outputs.

To generate adversarial examples, we select the top- $k$  influential words in each sentence from the test set, measured by  $\ell_2$ -norm of IS, and replace them with their synonyms with regard to different embedding methods. To ensure the generation of adversarial pair, we set a lower bound for the TME semantic distance between the original word  $\sigma$  and the selected synonym  $\sigma'$ , that is,  $d_T(\sigma, \sigma') \geq 0.01$ . For Glove, we simply replace the top- $k$  influential words with their synonyms. For TME, we leverage the Adversarial Pair

Embeddings	QC		Toxic	
	$k = 1$	$k = 2$	$k = 1$	$k = 2$
Glove	0.17	0.22	0.06	0.11
Weak Adversarial Pairs	0.34	0.46	0.11	0.23
Strong Adversarial Pairs	<b>0.44</b>	<b>0.59</b>	<b>0.15</b>	<b>0.25</b>

Table 6: Comparison results of Attack Success Rate (ASR) with different embedding methods.

under two settings with  $d_S(\sigma, \sigma') \leq 0.15$ ,  $d_S(\sigma, \sigma') \leq 0.18$ , respectively, to generate adversarial examples. Here, when  $\epsilon$  is set to 0.15, the constraint on semantic distance for natural language is relatively strict. The resulting adversarial samples are semantically clear and have minor changes compared to the original sentences. We denote this kind of adversarial pair as *Weak Adversarial Pairs*. When  $\epsilon$  is set to 0.18, however, the constraint on semantic distance becomes more relaxed. The resulting sentences have more different semantics and there might be some local grammar issues, but still can be classified into the same label as the original ones. We refer to this type of adversarial pair as *Strong Adversarial Pairs*. For Toxic, we set  $\epsilon$  to 0.12 and 0.2 for Weak Adversarial Pairs and Strong Adversarial Pairs, respectively. The comparison results are shown in Table 6. We can see that using adversarial pairs guided by TME achieves higher ASR than using Glove. For example, on QC news dataset, we’ve gained an average increase of 21% for Weak Adversarial Pairs and 32% for Strong Adversarial Pairs. The evaluation results validate the effectiveness of TME in capturing the decision logic and vulnerability of the target RNN.

#### 5.4. Discussion

*Computational Complexity.* The time complexity of the whole workflow is analyzed as follows. Suppose that the set of training samples  $\mathcal{D}_0$  has  $N$  words in total and its alphabet  $\Sigma$  contains  $n$  words, and is augmented as  $\mathcal{D}$  with  $t$  epochs (i.e. each sentence in  $\mathcal{D}_0$  is transformed to  $t$  new sentences in  $\mathcal{D}$ ), hence  $|\mathcal{D}| = (t + 1)N$ . Assume that a probabilistic output of RNNs is a  $m$ -dim vector, and the abstract states set  $\hat{S}$  contains  $k$  states.

To start with, the augmentation of  $\mathcal{D}_0$  and tracking of probabilistic outputs in  $\mathcal{D}$  will be completed in  $\mathcal{O}(|\mathcal{D}|) = \mathcal{O}(t \cdot N)$  time. Besides, the time complexity of k-means clustering algorithm is  $\mathcal{O}(k \cdot |\mathcal{D}|) = \mathcal{O}(k \cdot t \cdot N)$ . The count of abstract transitions will be done in  $\mathcal{O}(n)$ . As for the processing of transition matrices, we need to calculate the transition probability for each word  $\sigma$  with each source state  $\hat{s}_i$  and destination state  $\hat{s}_j$ , which costs  $\mathcal{O}(k^2 \cdot n)$  time. Finally, the context-aware enhancement on transition matrices takes

$\mathcal{O}(k \cdot n)$  time.

Note that  $\mathcal{O}(n) = \mathcal{O}(N)$ , hence we can conclude that the time complexity of our whole workflow is  $\mathcal{O}(k^2 \cdot t \cdot N)$ . So the time complexity of our approaches only takes linear time w.r.t. the size of the dataset, which provides theoretical extraction overhead for large-scale data applications.

For the explanation analysis, the IS score can be computed in constant time, as this process simply involves multiplying two matrices. Assuming the vocabulary contains a total of  $n$  words, and a sequence  $s$  with  $k$  words, conducting an adversarial attack on this sequence requires  $\mathcal{O}(k + m \log k)$  time to find the top- $m$  influential words, and it costs  $\mathcal{O}(n)$  time to find an optimal adversarial pair in the entire vocabulary through enumeration. Thus, if  $m$  words need to be replaced, the time complexity for the entire process is  $\mathcal{O}(k + m \log k + nm)$ .

*Applicability to other RNNs.* Although the proposed framework for WFA extraction and explanation of RNNs is customized for natural language tasks, we point out that some of its components can be generalized to other types of RNNs as well.

First, the identified transition sparsity and context-awareness problems in WFA extraction for natural language tasks may also occur in RNNs used in other domains, thus the proposed methods to address these problems are applicable to them as well. Thus, the empirical method to complement the missing rules in the transition diagram and the adjustment of transition matrices to enhance the context-awareness of the WFA can also be applied to other types of RNNs. However, the data augmentation tactics proposed in the paper may need to be adapted to suit the specific characteristics of other types of RNNs. Specifically, we can perform data augmentation on natural language samples as long as the synthetic sentences make sense. However, other datasets, such as formal languages, do not possess this property.

As for the explanation analysis, the application of our method for RNN explanation is not limited to natural language tasks. As long as a WFA can be extracted from the target RNN, the method for explanation is applicable. In fact, our study on RNN explainability only involves the extraction of vector representation of words through the transition matrices of the WFA, thus this component of our framework is highly generalizable and can be applied to various other domains beyond natural language processing.

## 6. Related Work

Many research efforts have been made to understand the behaviours of RNNs with an extracted model. We discuss the extraction methods and their applications respectively in the following.

### 6.1. Model Extraction of RNNs

As Jacobsson reviewed in [26], the extraction approach of RNNs can be divided into two categories: pedagogical approaches and compositional approaches.

*Pedagogical Approaches.* Many research works consider using pedagogical approaches to abstract RNNs by leveraging explicit learning algorithms, such as the  $L^*$  algorithm [7]. Earlier works date back to two decades ago, when Omlin et al. attempted to extract a finite model for Boolean-output RNNs [27; 28; 29]. Recently, Weiss et al. [8] proposed an approach to extracting DFA from RNN-acceptors based on  $L^*$  algorithm. Later, they presented a weighted extension of  $L^*$  algorithm that extracted probabilistic deterministic finite automata (PDFA) from RNNs [9]. Okudono et al. [10] proposed a weighted extension of  $L^*$  algorithm to extract WFA for real-value-output RNNs. Overall, pedagogical approaches have achieved great success in abstracting RNNs for small-scale languages, particularly formal languages. Such exact learning approaches have intrinsic limitation in scalability w.r.t. the language complexity, thus are not suitable for automata extraction for natural language models.

*Compositional Approach.* Another technical line for automata extraction from RNNs is the compositional approach, which leverages unsupervised algorithms (e.g. k-means, GMM) to cluster state vectors as abstract states [30; 31]. Wang et al. [11] studied the key factors in the compositional approach that influence the reliability of the extraction process, and proposed an empirical rule to extract DFA from RNNs. Later, Zhang et al. [21] followed the state encoding of compositional approach and proposed a WFA extraction approach from RNNs, which can be applied to both grammatical languages and natural languages. In this paper, the proposed WFA extraction approach from RNNs also falls into the line of compositional approach, but aims at proposing transition rule extraction method to address the transition sparsity problem and enhance the context-aware ability, which is customized for natural language tasks.

### 6.2. Model-based RNN Analysis and Explanation

There are a series of works focusing on deriving the extracted models for further applications, where the abstract models are more amenable to analysis and explanation.

*Model-based Analysis.* Model extraction techniques have been widely used to aid the analysis of RNNs, since the extracted models can be regarded as an approximation of the target RNNs, on which are easier to operate and perform analysis. [13] is a representative work for model-based RNN analysis, which leverages the extracted model to detect adversarial examples and increase test coverage of the target RNNs. Later, [15] proposed a model-based approach for robustness analysis of RNNs. Xie et al. [16] proposed to leverage the extracted model to identify buggy behaviors and further for automatic repairment of RNNs. In this paper, based on the extracted WFA, we proposed a new embedding method TME, which provides a new insight on RNN analysis for natural language tasks. With the proposed contrastive pairs derived by TME, we can analyze task-oriented semantics of the target RNNs, which further can be applied to boost pretraining and adversarial example generation for RNNs.

*Model-based Explanation.* There are also several works devoted to explaining the mechanism of RNNs with the aid of surrogate models. Krakovna et al. [32] presented an interpretation method for RNNs by using hidden markov models (HMMs) to simulate the source RNNs. Hou et al. [33] proposed an approach to interpreting the effect of gates on the mechanism of RNNs by using the extracted finite state automata. Jiang et al. [34] proposed a hybrid model *FA-RNNs*, which is trainable, generalizable as well as interpretable. There are also works operating directly on the structure of RNNs. Guo et al. [35] proposed an interpretable LSTM neural network equipped with tensorized hidden states, which could learn variable-specific representations. In this work, by leveraging the extracted WFA, we proposed a global explanation method, which computes the word-wise influence score on RNN decisions, and a contrastive explanation method, where the identified collaborative and adversarial repairs effectively characterize the task-oriented semantics learned by the target RNN.

## 7. Conclusion

In this paper, we propose a general framework for weighted automata extraction and explanation of RNNs for natural language tasks. We introduce a novel approach to extracting transition rules of weighted finite automata from recurrent neural networks. In particular, we address the transition sparsity problem and complement the transition rules of missing rows, which effectively improves the extraction precision. In addition, we present an heuristic method to enhance the context-aware ability of the extracted WFA. We further propose two augmentation tactics to track more transition behaviours of

RNNs. Both theoretical analysis and experimental results demonstrate the efficiency and precision of our rule extraction approach for natural language tasks. Based on the extracted model, we propose a word embedding method, Transition Matrix Embeddings (TME), to construct task-oriented explanations of the target RNN, including a word-wise global explanation method of RNNs, and a contrastive method to interpret the word semantics that the RNNs learned from the task.

### *Acknowledgements*

This work was sponsored by the National Natural Science Foundation of China under Grant No. 62172019, 61772038, and CCF-Huawei Formal Verification Innovation Research Plan; partially funded by the ERC under the European Union’s Horizon 2020 research and innovation programme (FUN2MODEL, grant agreement No. 834115).

### **References**

- [1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
- [2] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, D. Yu, Convolutional neural networks for speech recognition, *IEEE/ACM Transactions on audio, speech, and language processing* 22 (10) (2014) 1533–1545.
- [3] Y. Goldberg, Neural network methods for natural language processing, *Synthesis lectures on human language technologies* 10 (1) (2017) 1–309.
- [4] Z. Che, S. Purushotham, K. Cho, D. Sontag, Y. Liu, Recurrent neural networks for multivariate time series with missing values, *Scientific reports* 8 (1) (2018) 1–12.
- [5] R. Wang, Z. Li, J. Cao, T. Chen, L. Wang, Convolutional recurrent neural networks for text classification, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–6.
- [6] D. Datta, P. E. David, D. Mittal, A. Jain, Neural machine translation using recurrent neural network, *International Journal of Engineering and Advanced Technology* 9 (4) (2020) 1395–1400.
- [7] D. Angluin, Learning regular sets from queries and counterexamples, *Information and computation* 75 (2) (1987) 87–106.

- [8] G. Weiss, Y. Goldberg, E. Yahav, Extracting automata from recurrent neural networks using queries and counterexamples, in: *International Conference on Machine Learning*, PMLR, 2018, pp. 5247–5256.
- [9] G. Weiss, Y. Goldberg, E. Yahav, Learning deterministic weighted automata with queries and counterexamples, in: *Advances in Neural Information Processing Systems*, Vol. 32, Curran Associates, Inc., 2019, pp. 8558–8569.
- [10] T. Okudono, M. Waga, T. Sekiyama, I. Hasuo, Weighted automata extraction from recurrent neural networks via regression on state spaces, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI Press, 2020, pp. 5306–5314.
- [11] Q. Wang, K. Zhang, A. G. Ororbia II, X. Xing, X. Liu, C. L. Giles, An empirical evaluation of rule extraction from recurrent neural networks, *Neural computation* 30 (9) (2018) 2568–2591.
- [12] Q. Wang, K. Zhang, X. Liu, C. L. Giles, Verification of recurrent neural networks through rule extraction, *arXiv preprint arXiv:1811.06029*.
- [13] X. Du, X. Xie, Y. Li, L. Ma, Y. Liu, J. Zhao, Deepstellar: Model-based quantitative analysis of stateful deep learning systems, in: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 477–487.
- [14] G. Dong, J. Wang, J. Sun, Y. Zhang, X. Wang, T. Dai, J. S. Dong, X. Wang, Towards interpreting recurrent neural networks through probabilistic abstraction, in: *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 2020, pp. 499–510.
- [15] X. Du, Y. Li, X. Xie, L. Ma, Y. Liu, J. Zhao, Marble: Model-based robustness analysis of stateful deep learning systems, in: *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, 2020, pp. 423–435.
- [16] X. Xie, W. Guo, L. Ma, W. Le, J. Wang, L. Zhou, Y. Liu, X. Xing, Rnnrepair: Automatic rnn repair via model-based analysis, in: *International Conference on Machine Learning*, PMLR, 2021, pp. 11383–11392.
- [17] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural computation* 9 (8) (1997) 1735–1780.

- [18] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.
- [19] D. M. W. Powers, Applications and explanations of zipf’s law, in: New methods in language processing and computational natural language learning, 1998, pp. 151–160.
- [20] X. Li, D. Roth, Learning question classifiers, in: COLING 2002: The 19th International Conference on Computational Linguistics, 2002.
- [21] X. Zhang, X. Du, X. Xie, L. Ma, Y. Liu, M. Sun, Decision-guided weighted automata extraction from recurrent neural networks, in: Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI), AAAI Press, 2021, pp. 11699–11707.
- [22] Jigsaw, Toxic comment classification challenge, <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge> Accessed April 16, 2022.
- [23] M. Menéndez, J. Pardo, L. Pardo, M. Pardo, The jensen-shannon divergence, Journal of the Franklin Institute 334 (2) (1997) 307–318.
- [24] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781.
- [25] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of machine learning research 9 (11).
- [26] H. Jacobsson, Rule extraction from recurrent neural networks: Ataxonomy and review, Neural Computation 17 (6) (2005) 1223–1263.
- [27] C. Omlin, C. Giles, C. Miller, Heuristics for the extraction of rules from discrete-time recurrent neural networks, in: [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, Vol. 1, IEEE, 1992, pp. 33–38.
- [28] C. W. Omlin, C. L. Giles, Extraction of rules from discrete-time recurrent neural networks, Neural networks 9 (1) (1996) 41–52.
- [29] C. W. Omlin, C. L. Giles, Rule revision with recurrent neural networks, IEEE Transactions on Knowledge and Data Engineering 8 (1) (1996) 183–188.

- [30] Z. Zeng, R. M. Goodman, P. Smyth, Learning finite state machines with self-clustering recurrent networks, *Neural Computation* 5 (6) (1993) 976–990.
- [31] A. L. Cechin, D. Regina, P. Simon, K. Stertz, State automata extraction from recurrent neural nets using k-means and fuzzy clustering, in: 23rd International Conference of the Chilean Computer Science Society, 2003. SCCC 2003. Proceedings., IEEE, 2003, pp. 73–78.
- [32] V. Krakovna, F. Doshi-Velez, Increasing the interpretability of recurrent neural networks using hidden markov models, arXiv preprint arXiv:1606.05320.
- [33] B.-J. Hou, Z.-H. Zhou, Learning with interpretable structure from gated rnn, *IEEE transactions on neural networks and learning systems* 31 (7) (2020) 2267–2279.
- [34] C. Jiang, Y. Zhao, S. Chu, L. Shen, K. Tu, Cold-start and interpretability: Turning regular expressions into trainable recurrent neural networks, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 3193–3207.
- [35] T. Guo, T. Lin, Y. Lu, An interpretable lstm neural network for autoregressive exogenous model, arXiv preprint arXiv:1804.05251.