特征变化下循环神经网络的可解释性研究

侯博建[1),2)] 姜远[1),2)]

[1)](计算机软件新技术国家重点实验室(南京大学) 南京 210023)

[2)](软件新技术与产业化协同创新中心(南京大学) 南京 210023)

摘 要 循环神经网络（RNN）已经成功应用在诸多领域，比如语音识别、文本分类等。但是他们的内在工作机制尚没有被很好的解释。在本文中，我们关注循环神经网络（RNN）并尝试研究其可解释性。我们发现处理序列数据的有限状态自动机（FSA）具有更多可解释的内部机制，并可以从 RNN 中学得。另外，在开放环境中，特征经常发生变化，因此我们也需要将这种特征改变的情况融入我们的设定中。我们提出了一种从 RNN 学习 FSA 的方法，并研究了特征变化带来的影响。我们首先分析了 RNN 的性能如何受到特征变化和门的数量的影响，然后给出了人类可以模拟从而易于理解的 FSA 的图示、分析了以数值计算为依托的隐层节点跳转背后的语义。我们的结果表明，具有简单门控结构的 RNN（例如最小门控单元（MGU））是更理想的 RNN 结构，并且 FSA 中导致特定分类结果的跳转的词具有相同词性，这在语义层面是更容易让人理解的。此外，我们发现不同的特征不会对性能产生严重影响，这表明我们为了减少时间和空间复杂度而应该选择较短的特征。

关键词 机器学习；循环神经网络；有限状态自动机；可解释性；特征变化

Learning Interpretability from RNN with Feature Changing

HOU Bo-Jian[1),2)] JIANG Yuan[1),2)]

[1)](National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023)

[2)](Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210023)

Abstract Recurrent Neural Networks (RNNs) have been successfully applied to many practical tasks, such as speech recognition, text classification, etc. However, their inner working mechanisms are not clearly understood. In this paper, we focus on Recurrent Neural Networks (RNNs) and attempt to learn interpretability from it. We find that Finite State Automaton (FSA) that processes sequential data has more interpretable inner mechanism and can be learned from RNNs. Besides, in open environment, the features often change and thus we need to incorporate this situation into our setting. In our work, we propose a method to learn FSA from RNN and study the impact of feature changing. From the FSA's point of view, we analyze how the performance of RNNs are affected by feature changing and the number of gates. We then give the graphical illustration of FSA for human beings to simulate, which shows the interpretability. With the graphical illustration, we also analyze the semantic meaning behind the transition of numerical hidden states. Our results suggest that RNNs with simple gated structure such as Minimal Gated Unit (MGU) is more desirable and the words on the transitions in FSA leading to specific classification result are within the same class, which are understandable by human beings semantically. Furthermore, we find that different features will not make a serious difference to the performance, which suggest us to choose a short one for the sake of saving time and space complexity.

收稿日期：2019年8月13日；最终修改稿收到日期：2020年2月-日。本课题得到国家自然科学基金项目(资源受限的多视图学习技术研究, 61673201)的资助。侯博建，男，1992年生，博士研究生，主要研究领域为人工智能、机器学习和数据挖掘。E-mail: houbj@lamda.nju.edu.cn。姜远（**通信作者**），女，1976年生，博士，教授，中国计算机学会(CCF)会员(10759M)，主要研究领域为人工智能、机器学习和数据挖掘。E-mail: jiangyuan@nju.edu.cn。

Key words machine learning; recurrent neural network; finite state automata; interpretability; feature changing

1 引言

现如今，循环神经网络[1]（Recurrent Neural Network，以下简称 RNN）技术已经取得了很大的成功，但是依然存在很多不足和亟待解决的问题，比如模型不容易解释、很难适应环境变化、数据量不够等。学件（learnware）[2]针对这些问题强调一个好的模型应该是可解释的（comprehensible 或者 interpretable）、可发展变化的（evolvable）和可重用的（reusable）。同时研究这三个问题比较困难，所以本文先尝试同时研究前两者，也即 RNN 的可解释性和可发展变化性。

可解释性还没有一致的定义，但已经有一些比较合理的定义值得一提。例如 Kim 等人在 2016 年提出了一个大家比较认可的定义[3]：可解释性是人类可以和模型预测一致的程度。另外一个被广泛接受的定义是由 Miller[4]提出的：可解释性是人类理解模型做相应决策的原因的程度。Lipton 在 2016 年也提出了一个可解释性的具体定义[5]，也即可模拟性（human-simulability）：一个具有可解释性的模型，人类是可以模拟其运行过程的。更确切来说，如果一个模型可模拟，那么人类输入数据和此模型的参数，能够在合理时间范围内计算每一步骤，最终获得和模型一致的预测结果。这三种定义其实是一致的：如果一个人可以模拟某个模型的运行过程(Lipton)，那么这个人就可以和模型的预测结果一致(Kim)，也可以理解这个模型做出相应决策的原因(Miller)，所以我们关注可模拟性。

可发展变化性指的是模型可以适应环境的变化：在开放环境中数据的分布、特征和类别都会经常发生变化。我们关注和研究比较常发生的特征的变化[6]，而同时考虑可解释性和特征变化就很自然地想到去研究特征变化对 RNN 可解释性的影响。

从深度模型中获取可解释性通常来说是困难的，但是对于循环神经网络却有解决的办法。作为深度神经网络的主要成员 RNN，尤其是带有门控机制的 RNN，比如带有一个门的 MGU[7]、带有两个门的 GRU[8]和带有三个门的 LSTM[9]在学习*序列数据*上都有着丰富的应用，比如语音识别[10]、给图片加标题[11]、情感分析[12]等。而除了 RNN，有限状态自动机（英文 Finite State Automata，缩写为 FSA）[13],[14],[15]也是具有处理*序列数据*能力的工具。FSA 主要包含有限的状态和状态之间的转移，它会根据外界的序列输入在状态之间跳转。FSA 这样的状态跳转过程和 RNN 在接收序列输入后从一个隐层跳转到下一个隐层的过程很相似；而不同之处在于 FSA 的内部工作机制可以被人类模拟，这符合前文所说的可模拟性，从而具有了更好的可解释性。而 RNN 内部工作机制却由于全是数值上的计算而很难让人明白里面到底发生了什么。因此 FSA 的这一特性启发我们从 RNN 中学得 FSA，用 FSA 的天然可以理解的特性来尝试理解 RNN 的内部机理。

为了从 RNN 中学得 FSA 并使用它来理解 RNN 的内部机理，我们需要回答两个问题：一个是*怎么学*，一个是到底要*理解哪些内容*。对于第一个问题，我们了解到传统的不带门控机制的 RNN 的隐层节点的输出是有聚类特质的[16],[17]，我们可以利用这一特质来学得 FSA，但是我们不知道这样的聚类特质是否在带有门控机制的 RNN 内部依然存在；并且我们还要考虑效率问题，因为带有门控机制的 RNN 经常处理大规模的数据。对于第二个问题，我们需要分析门在这些门控机制的 RNN 内部起到了多大的作用，尤其是不同的数量的门会有怎样不同的影响；再者，FSA 内状态之间的跳转是有物理含义的，或许可以从这个角度来获取 RNN 内部隐层之间跳转所蕴含的语义信息，从而更好的理解 RNN 的内部机理；另外，在开放动态环境下，数据的特征可能会发生变化[6]，可能会对 RNN 的可解释性造成影响，因此我们还要考虑和分析特征变化对 RNN 的可解释性的影响。

本文首先验证了带有门控机制的 RNN 的隐层节点的输出同样具有聚类的特质，然后用高效的 k-means++[18]来对这些隐层节点的输出做聚类。虽然我们使用了更先进的聚类方法，但是聚类方法并不是本文的核心问题，只要能在多项式时间内做有效的聚类都是可行的。聚类后我们把隐层节点的类簇作为状态，设计出五个必要元素来学得 FSA。这五个元素分别是词汇集合、状态集合、开始状态、接受状态集合和状态转换函数。从 RNN 学得 FSA 之后，我们便可以画出它们的图示，从而通过模拟其运行过程来方便理解内部的运行机理，我们同样也可以用其来进行分类。在人工数据集和真实数据集上的分类结果表明 RNN 与其学得的 FSA 在性能

上具有统一性，从而表明 FSA 在一定程度上能够代表 RNN，那么在一些需要可解释性或者信任的场景下比如医疗诊断、核电站决策等，FSA 就有机会替代 RNN 了。另外，实验结果表明简单门控结构的 RNN（例如最小门控单元（MGU））是更理想的 RNN 结构，并且 FSA 中导致特定分类结果的跳转的词具有相同词性，这在语义层面也是容易让人理解的。此外，我们发现不同的特征不会对性能产生严重影响，这表明我们为了减少时间和空间复杂度而应该选择较短的有效特征。

2 背景知识

本章将介绍一个不带门控机制的经典 RNN 以及三个带有门控机制的 RNN，并根据这些 RNN 的内部结构来进一步讨论可解释性这一话题，最后是介绍相关工作。

不带有门控机制的经典 RNN 早在上个世纪九十年代就由文献[1]提出了。顾名思义，这个不带有门控机制的经典 RNN 是没有任何门的，结构也非常简单，并且只适用于小数据集，所以我们又称它为简易 RNN（英文 Simple RNN，缩写为 SRN）。总体而言，SRN 遍历序列里的每一个元素并将其作为输入，然后结合*当前的输入*和*上一步隐层节点的输出*来计算当前隐层节点的输出。更具体来说，在时刻 t，我们把序列的第 t 个元素 $\mathbf{x}_t$ 输入到隐层节点。隐层节点会根据当前的输入 $\mathbf{x}_t$ 和上一步隐层节点的输出 $\mathbf{h}_{t-1}$ 根据以下的公式计算出当前的输出 $\mathbf{h}_t$:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t).$$

这里的 f 通常是线性函数和非线性激活函数的复合，如下所示：

$$\mathbf{h}_t = \tanh(W[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b})$$

其中矩阵 W 是与 $\mathbf{h}_{t-1}$ 和 $\mathbf{x}_t$ 紧密相关的参数，$\mathbf{b}$ 是有偏项。SRN 的任务就是学得 W 和 $\mathbf{b}$。

虽然 SRN 能够较为有效地处理序列数据，但是随着互联网时代的发展我们面对的数据的规模变得越来越大，SRN 简单的结构已经无法处理这么大规模的数据了，所以我们需要更复杂更深的模型来应对这一情况[19],[20]。但是直接扩展 SRN，使得其更深的话，会面临梯度消失或者爆炸的问题，从而使得学习 SRN 变得困难[9],[21]。不过幸运的是，为了解决梯度消失或者爆炸的问题，人们在 RNN 内引入各种各样的门来控制隐层节点内信息的流动，带门控机制的 RNN 应运而生。这其中，长短时记忆模型（英文 Long Short Term Memory，缩写为 LSTM）和门控循环单元（英文 Gated Recurrent Unit，缩写为 GRU）是时下主流的两种带门控机制的 RNN。LSTM 有三个门，包括一个输入门（英文 output gate），一个遗忘门（英文 forget gate）和一个输出门（英文 output gate）。输入门用来控制新信息的添加，遗忘门决定要记住多少旧的信息，而输出门是用来控制当前信息的输出。GRU 有两个门，一个叫更新门（英文 update gate），一个叫重置门（英文 reset gate）。更新门和 LSTM 的遗忘门类似，用来控制遗忘多少信息，而重置门和 LSTM 的输入门类似，用来控制添加多少新的信息。

表 1 三种带门控机制的 RNN 的运算方式。加粗的是向量。

MGU（最小门控单元）	
$\mathbf{f}_t = \sigma(W_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$	（门）
$\mathbf{s}_t = \tanh(W_h[\mathbf{f}_t \text{ e } \mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_h)$	
$\mathbf{h}_t = (1-\mathbf{f}_t) \text{ e } \mathbf{h}_{t-1} + \mathbf{f}_t \cdot \mathbf{s}_t$	
GRU（门控循环单元）	
$\mathbf{z}_t = \sigma(W_z[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_z)$	（门）
$\mathbf{r}_t = \sigma(W_r[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_r)$	（门）
$\mathbf{s}_t = \tanh(W_h[\mathbf{r}_t \text{ e } \mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_h)$	
$\mathbf{h}_t = (1-\mathbf{z}_t) \text{ e } \mathbf{h}_{t-1} + \mathbf{z}_t \cdot \mathbf{s}_t$	
LSTM（长短时记忆模型）	
$\mathbf{f}_t = \sigma(W_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$	（门）
$\mathbf{i}_t = \sigma(W_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$	（门）
$\mathbf{o}_t = \sigma(W_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$	（门）
$\mathbf{s}_t = \tanh(W_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c)$	
$\mathbf{c}_t = \mathbf{f}_t \text{ e } \mathbf{c}_{t-1} + \mathbf{i}_t \text{ e } \mathbf{s}_t$	
$\mathbf{h}_t = \mathbf{o}_t \text{ e } \tanh(\mathbf{c}_t)$	

LSTM 和 GRU 都在隐层节点添加了若干个门，产生了很多额外的参数需要调整和计算，所以并不够高效。为了解决这个问题，文献[7]提出了最小门控单元（英文 Minimal Gated Unit，缩写为 MGU）。MGU 虽然只有一个遗忘门，却和 LSTM、GRU 有着不相上下的性能。所以总体来说相比于 LSTM 和 GRU，MGU 具有更简单的结构、更少的参数和更快的训练和调参过程等优点。

SRN、MGU、GRU 和 LSTM 的数学形式化表示都总结在表 1 中，其中

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

是 S 型生长曲线函数（此函数作用在输入向量的每

一维上），e 的作用是使得两个向量上对应的元素分别相乘。表 1 中所有 RNN 的门都用“(门)”标记出来了，其中可以很清楚地看到 MGU 有一个门，GRU 有两个门，而 LSTM 有三个门。为了更清楚地展示这些带有门机制的 RNN 是如何计算的，图 1 还给出了它们的内部数据流动和计算图。

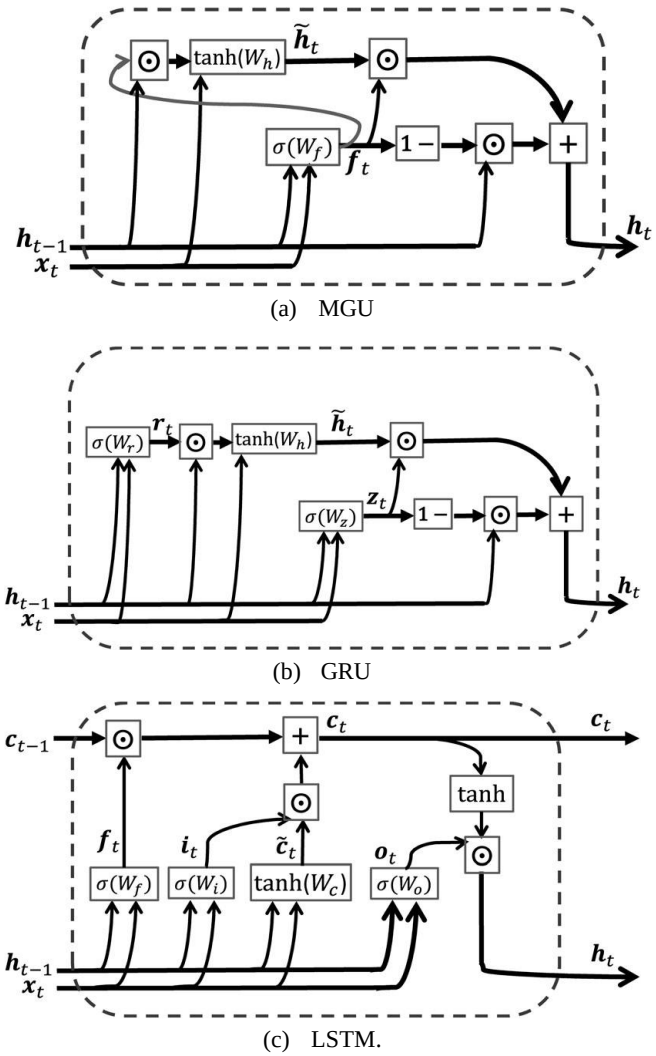


图 1 带有不同门机制的 RNN 内部数据流动和计算图[7]。

尽管这些门一定程度上解决了梯度消失和爆炸的问题，但是带有门控的机制使得这些 RNN 的内部机理过于复杂，从而给对它们的可解释性研究带来困难。总的来说有三个主要因素导致了这些带有门控机制的 RNN 内部机理的复杂性。一个是它从经典 RNN[22]继承过来的循环结构：尽管循环结构被认为是处理序列数据的关键，但是相同的神经单元循环地处理不同的输入会让人们对这种分类的过程产生困惑。另外一个复杂性来自这些门本身：尽管人们青睐 MGU 的原因是它比 LSTM 和 GRU 有着更少的门，但是门的作用并没有被完全理解，尤其是到底多少个门就足够了这样的问题依然不清楚。第三个复杂性来自于带有门控机制的 RNN 的内部过程是基于数值运算的，然而人们没有办法直接把数值向量和具体的现实含义联系到一起，所以这些数值运算和数值向量也很让人费解。总之，带有门控机制的 RNN 复杂的内部机理很难让人理解它，可解释性也就无从说起了。

最近有两个具有代表性的理解 RNN 的工作被提出。一个是由 Karpathy 等人在 2015 年提出的专注于可视化输入特征的工作[23]。这个工作使用热图(heat map)来显示出输入的文本中哪些部分和特定的结构是相关的，比如在括号中的会被自动标为红色，在括号外的会被自动标为蓝色。另一个工作[8]使用“t 分布随机邻接嵌入”（英文 t-Distributed Stochastic Neighbor Embedding，缩写为 t-SNE）[24]这个降维工具来可视化地展示通过“RNN 编码解码模型”学得的短语表示在二维空间上的关系。但是它们都没有考虑 RNN 隐层节点之间的关系，因此 RNN 的内部机理依然很难理解和解释。此外，还有一些关注深度神经网络的另一分支卷积神经网络（CNN[19]）的解释性工作，比如文献[25]、[26]、[27]、[28]等着重于可视化；文献[29]和[30]等关注网络内部节点之间关系的工作。但是这些理解或解释 CNN 的工作并不能直接迁移到 RNN 模型上。

此外值得一提的是本文受到一些 RNN 规则抽取和 RNN 可解释性的工作[16],[17],[31]的启发。它们也从 RNN 中学得有限状态机来研究语法规则或者 RNN 的可解释性，但是它们并没有考虑特征变化对 RNN 可解释性的影响。

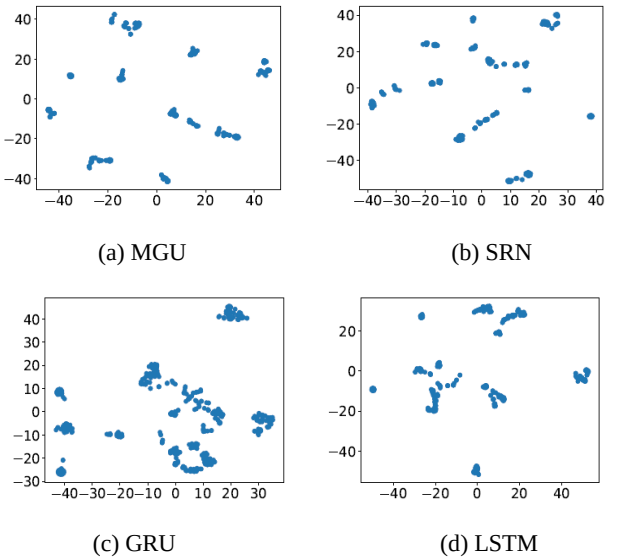
本文从 RNN 中学得有限状态自动机（英文 Finite State Automata，缩写 FSA）来探索在特征变化下带有门控机制的 RNN 的可解释性。我们发现 MGU 虽然拥有最少个数的门，但仍然能够在 FSA 的角度下比其他 RNN 具有更好的性能。这可以为设计更好的 RNN 做出一些指导。我们还发现 FSA 和 RNN 具有统一性，所以 FSA 有机会在需要取得人类信任的任务中替代 RNN。另外，不同的特征不会对 FSA 的性能产生严重影响，这表明我们为了减少时间和空间复杂度应该选择较短的有效特征。

3 学习有限状态自动机

本章主要介绍如何学习 FSA，并通过学得的

FSA 来研究循环神经网络的可解释性。第一节会介绍我们方法的框架，第二节是具体的方法介绍。

3.1 方法框架



(a) MGU (b) SRN

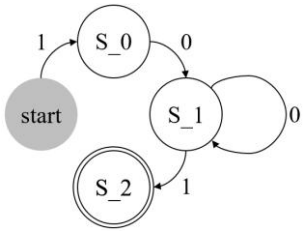
(c) GRU (d) LSTM

图 2 隐层节点输出降到 2 维的图示，它们具有聚类特质。

文献[16]告诉我们当不带门机制的 SRN 的隐层节点的输出 $\mathbf{h}_t$ 被当做高维空间的一个点时，很多输入到 SRN 中的序列所对应的 $\mathbf{h}_t$ 不会没有规律地散落，而是会倾向于聚类，我们发现带有门机制的 RNN 也具有这样的特性。为了有一个直观的感受，我们使用“t 分布随机邻接嵌入”（英文 t-Distributed Stochastic Neighbor Embedding，缩写为 t-SNE）[24]工具把 400 个维度为 10 的 $\mathbf{h}_t$ 降维到 2 维，然后把他们画在了二维平面上。图 2 的(a)到(d)分别给出了 MGU，SRN，GRU 和 LSTM 的 400 个隐层节点的输出 $\mathbf{h}_t$ 的分布。从图中可以看到，四种 RNN 的隐层节点输出 $\mathbf{h}_t$ 都体现出了聚类的特性。我们假设不同的类簇代表不同的*状态*，而当序列里的一个元素输入到 RNN 之后，隐层节点的输出发生了变化，那么其对应的类簇也即*状态*发生了转移，所以神经网络表现的就像是一个状态机。我们假设状态是有限的，那么我们就可以从 RNN 里学得一个 FSA 了。

学习 FSA 主要有三步。第一步我们首先在训练集上训练一个 RNN，然后第二步在验证集 V 上获得所有的隐层节点输出集合 H，并对其做聚类。最后第三步通过对 H 聚类来学得 FSA。在获得 FSA 之后，我们可以通过它来直观地看到 RNN 的内部运作从而分析 RNN 的可解释性。第一步训练 RNN，我们使用和文献[7]相同的策略，所以这里就不展示细节了。第二步对 H 做聚类，我们直接使用 k-means 的一个更鲁棒的版本 k-means++[18]。它能够更智能地初始化类簇中心，使得聚类结果一定是收敛的，从而保证了最后结果的稳定性。接下来我们将着重介绍第三步对 FSA 的学习过程。

3.2 学习FSA



(a) FSA 图示

状态	0	1
start		S_0
S_0	S_1	
S_1	S_1	S_2
S_2		

(b) FSA 状态转移表格

图 3 有限状态机（图示和及其状态转移表格）。

一个 FSA M 是一个 5 元组，包含五个元素，分别是 Σ、Q、R、F、δ。其中 Σ 是字母表，是在输入序列里出现过的元素的集合（如图 3 中的导致状态跳转的字符“0”和“1”）；Q 是状态集合（如图 3 中的状态“start”、“S_0”、“S_1”和“S_2”组成的集合）；$R \in Q$ 是开始状态（如图 3 中的状态“start”）；$F \subseteq Q$ 是接受状态的集合（如图 3 中的双环圆圈表示的状态“S_2”组成的集合）；$\delta : Q \times \Sigma \to Q$ 表示 M 中的状态转移（图 3(b)即是这种状态转移的等价表示）。当给 FSA 输入字母表 Σ 中的元素时，会引发 FSA 内部状态的跳转，输入结束时如果状态停在接受状态，则表明 FSA 判断输入的序列为正类，反之则为负类。在图 3 的例子中，当输入“1001”到 FSA 中时，最后会跳转到接受状态“S_2”，那么意味着图 3 中的 FSA 接受了“1001”，或者说 FSA 判定“1001”为正类。为了学得一个 FSA，我们将一一阐述如何学得 FSA 的这五个元素。

3.2.1 学习字母表 Σ

首先字母表 Σ 是容易从数据中学得的。例如，如果数据 D 是由自然语言构成的句子集合，那么 Σ 就等于所有的句子中所有单词的集合。那么我们有

$$\Sigma = \text{Vocabulary}(D), \tag{2}$$

其中 Vocabulary(D)指的是出现在 D 中的所有单词。

3.2.2 学习状态集合 Q

在训练 RNN 或者用 RNN 做测试时，每次我们从某个序列中输入一个元素到 RNN 中，在给定前一个隐层节点的输出 $\mathbf{h}_{t-1}$ 我们都能通过 RNN 的内部机制计算获得当前的隐层节点的输出 $\mathbf{h}_t$。而在 FSA 中，每次我们从 Σ 输入一个符号 s 到 FSA 中，根据

当前的状态和状态转移函数 $\delta: Q\times\Sigma\to Q$ 我们可以知道要转移到的下一个状态是什么。这两个过程很相似，所以我们可以把 RNN 中包含相似隐层节点的输出 **h** 的类簇作为 FSA 里的一个状态。那么状态集合 Q 可以被定义为

$$Q=\{C\,|\,\mathbf{h}\in C\}\cup\{R\},\tag{3}$$

其中 C 是隐层节点输出 **h** 的一个类簇。

3.2.3 学习接受状态集合 F

由于在开始时，输入元素到 RNN 中时，没有前一个隐层节点存在，所以我们定义开始状态 R 为不包含任何隐层节点输出的状态。那么开始状态 R 只是一个象征开始的符号。而接受状态 F 由类簇中心决定。由于 FSA 中的每一个状态都对应 RNN 中隐层节点输出的聚类簇，当我们用 RNN 的分类器来对类簇中心做分类，如果分类结果是正的，那么对应的状态就是接受状态。所以我们有

$$F=\{C\,|\,\mathrm{RNN}(\text{cluster center of }C)=1\},\tag{4}$$

其中 RNN(cluster center of C)表示用 RNN 对 C 的类簇中心做分类。

3.2.4 学习转移函数 δ

第五个元素状态转移函数 δ 是最难获得的。我们使用转移矩阵 $T\in[|Q|]^{|Q|\times|\Sigma|}$ 来表达状态转移函数 $\delta: Q\times\Sigma\to Q$。其中 $|Q|$ 表示 Q 中元素的个数，$[|Q|]$ 表示范围从 1 到 $|Q|$ 的整数集合，$|\Sigma|$ 表示 Σ 中元素个数。矩阵 T 的每一行代表一个状态（第一行代表开始状态 R，它在 Q 中的序号是 $|Q|$），T 的每一列代表 Σ 中一个符号 s。T 的第 i 行第 j 列元素 $T(i,j)$ 表示 FSA 在状态 i 并输入符号 s_j 时所转移到的下一个状态。为了得到 T，我们首先为每一个元素 s 计算一个跳转矩阵 N_s，其中第 i 行第 k 列元素表示在数据 D 所有的序列中，当输入 s 时从状态 i 跳转到状态 k 的频率，具体计算步骤如下：

1）给所有的类簇或者状态标上序号，在这个类簇中的隐层节点输出也跟其类簇序号对应；

2）遍历所有的隐层节点输出，当输入 s 造成从状态 i 到状态 k 的转移，则给 $N_s(i,k)$ 增加 1。

计算结束后，$N_s(i,k)$ 表示在输入 s 时状态从 i 跳转到 k 的次数。不过在输入 s 时，状态 i 可能会跳转到不同的状态。为了得到一个确定的 FSA 从而获得清晰的图示，我们在 N_s 的每一行只保留最大值，从而丢掉那些跳转频率小的值；当然这也会使得信息缺失从而造成 FSA 性能下降，不过这并不影响我们用 FSA 来研究 RNN 的可解释性。转移矩阵 T 就可以通过 N_s 来快速计算得到，其中 s_j 是 T 的第 j 列对应的字符：

$$T(i,j)=\arg\max_k N_{s_j}(i,k)\tag{5}$$

以上 FSA 的五个元素全部得到后，我们就学得了一个完整的 FSA，算法 1 总结了学习的所有步骤。

算法 1 学习有限状态自动机

输入：聚类个数 k

输出：一个有限状态自动机

1. 训练一个 RNN 模型，并在验证集 V 上做测试；
2. 对于 V 中的每一个序列里的每一个元素都记录下其对应的隐层节点输出，放到集合 H 里；
3.使用 k-means++对隐层节点输出集合 H 内的元素做聚类；
4.通过公式(2)获得字母表 Σ；
5.通过公式(3)获得状态集合 Q；
6.通过公式(4)获得接受状态集合 F；
7.针对 Σ 中每一个符号 s 计算一个跳转矩阵 N_s，$N_s(i,k)$ 表示在输入 s 时状态从 i 跳转到 k 的次数；
8.通过公式(5)获得转移矩阵 T，也即转移函数 δ。

4 实验测试

在这一章节，我们分别在人工数据集和真实数据集上做实验。在人工数据集上，我们从 RNN 中学得 FSA 后，将其内部的运行过程用图解的形式来展示，从而去理解 RNN 的内部机制。在第二个真实数据集上，我们主要探索不同的特征会带来什么样的影响，以及引导跳转的词汇的语义信息。

4.1 数据集和任务

我们在三个任务上进行实验。前两个任务是在人工数据集上做的模拟任务，第三个任务是在真实数据集上做的真实任务。

第一个任务是识别长度为 4 的 0/1 字符串是否为 0110，我们称此任务为“0110”。我们随机生成了 1000 个长度为 4 的 0/1 字符串作为训练集，生成了 16 个没有重复的长度为 4 的 0/1 字符串作为验证集（因为总共只有 16 种情况），并随机生成了 100 个长度为 4 的 0/1 字符串为测试集。第二个任务和

第一个任务相似，它是识别任意长度的 0/1 字符串里是否包含连续的 000。我们称此任务为“000”。由于序列的长度没有限制，所以样本空间是无限大的，这也导致这个问题的难度相比于第一个任务极大地提升了。我们随机生成了长度也随机的 3000 个 0/1 字符串。为了简化问题，限定字符串的长度为 5 到 20。同时用相同的方式还生成了 500 个样本作为验证集和 500 个样本作为测试集。

第三个任务是对来自 IMDB 的影评数据集[32]做情感分析，在这个数据集中，每一个示例都是对电影的一段评价，我们的任务就是去判断这些示例是正面的评价还是负面的评价。为了探索特征变化对性能以及可解释性的影响，我们使用长度或者种类不同的五种预训练好的词向量集来把影评中的每一个英文单词都映射为向量。其中包括 300 维的 word2vec[33]以及从 Wikipedia 2014 和 Gigaword 5 学得的 50 维、100 维、200 维和 300 维的 GloVe (Global Vectors)[34]向量。不同维度和种类的词向量就代表着输入 RNN 的示例的特征是不同的。探索特征变化的原因是一方面现实中会遇到特征变化的情况，另一方面也可以用来探索可解释性。可解释性可以从很多角度来理解，比如模型在什么时候会有效果，什么时候没有效果；为什么模型会产生这样的结果；怎么样设计模型可以有更好的性能表现。总结来说我们要“信任”模型的预测结果。所以探索性能与特征之间的关系可以让我们知道模型在什么时候有效果。这些都是探索可解释性的一部分。

对于这三个任务，我们研究的 RNN 都是在第二章提到的 MGU、SRN、GRU 和 LSTM。为了保证结果的可比性，我们把这四种 RNN 的隐层节点个数和隐层数都分别设置为 10 和 3。所有的实验都重复五次，并汇报平均结果，在表 4 中还汇报了五次结果的标准差。我们想强调的是选择“0110”、“000”和情感分析任务的数据集并不是随机的。对于人工数据集，在“0110”和“000”这两个难度相差很大的任务上我们获得了一致的结果和相同的结论，具体的内容在后面的章节中会讨论。所以对于识别不同的 0/1 序列（Tomita Grammars[35]），我们的例子是具有一般性的。而对于真实数据集，由于有限状态机（FSA）只能处理二分类任务，情感分析任务是我们最理想的选择：一方面文本作为序列化数据适合 RNN 处理，另一方面情感分析一般判断一段文字的情感是正面的还是负面的，为二分类任务，适合 FSA 来处理。但是这个例子至少在文本类任务上具有一般性。它向我们展示了 RNN 是如何处理带有语义的文本分类任务的。

4.2 聚类个数和特征变化的讨论

表 2 在任务“0110”上 FSA 达到 1 的准确率时聚类的个数。加粗的表示的是每一行最好的数据。

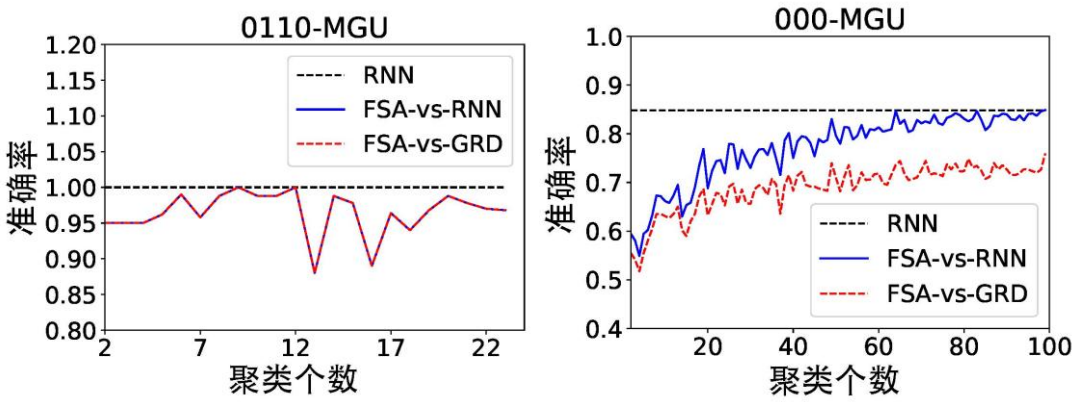
RNN 种类	MGU	SRN	GRU	LSTM
第一次	**5**	13	7	13
第二次	**8**	9	25	9
第三次	**6**	6	8	12
第四次	**5**	5	8	17
第五次	**6**	22	9	22
平均值	**6**	11	11.2	14.6

表 3 在任务“000”上 FSA 达到 0.7 的准确率时聚类的个数。加粗的表示的是每一行最好的数据。

RNN 种类	MGU	SRN	GRU	LSTM
第一次	38	84	201	**26**
第二次	**6**	28	109	72
第三次	**9**	28	201	20
第四次	**8**	41	85	19
第五次	**7**	180	201	22
平均值	**13.6**	72.2	159.4	31.8

根据我们的算法，在学得 FSA 时，我们需要对隐层节点做聚类，那么就需要设置类簇或者是 FSA 里状态的个数 k。越多的类簇也就意味着每个类簇里隐层节点越趋于减少。一个极端的例子是类簇个数和隐层节点个数相同，那么 FSA 内部状态的跳转就和 RNN 中隐层节点的转换相同了。所以当 k 足够大的时候，FSA 的性能就应该和 RNN 的性能相近。不过我们希望在保证性能的前提下 FSA 的状态个数越少越好，一方面可以避免过拟合、增加效率，还可以减少复杂度从而能够更好地被人模拟继而更容易被理解。所以如果在聚类个数很少的情况下还能获得好的分类效果，那说明 FSA 学得很好。

在任务“0110”中，我们设置聚类的个数从 2 到 64（因为长度为 4 的 0/1 序列只有 16 种，在验证集里我们只有 16 个互不相同的序列，所以总共有 64 个隐层节点输出）。表 2 给出了从 4 种 RNN 学得的 FSA 在分类准确率达到 100%（或 1）时的聚类个数。我们可以发现在五次实验中 MGU 总是最先达到 100%的准确率。平均来说，MGU 在聚类个数为 6 时准确率达到 100%，SRN 在聚类个数为



(a) MGU 在任务“0110”上的性能 (b) MGU 在任务“000”上的性能

图 4 模拟任务随着聚类个数的增加 FSA 的准确率，GRD 是真实标记（groundtruth）的简称。FSA-vs-RNN 指以 RNN 输出的标记作为衡量标准时 FSA 的准确率，FSA-vs-GRD 指以真实标记作为衡量标准时 FSA 的准确率。

11 时准确率达到 100%，GRU 在聚类个数为 11.2 时准确率达到 100%，LSTM 在聚类个数设置到 14.6 时准确率才达到 100%。

在任务“000”中，我们设置聚类的个数从 2 到 200。我们实际上有 500*l 个隐层节点，其中 l 是所有 500 个测试样本的平均长度，但是我们不需要这么多，因为我们的目标是观察 FSA 的性能在聚类个数较小的时候随着聚类个数的增多的变化情况。另外，由于这个任务比“0110”的任务困难，无论是原始 RNN 还是 FSA 都达不到 100%的准确率。所以我们把准确率阈值从 100%下调到 0.7。也即我们观察 FSA 首次达到 0.7 的准确率时聚类的个数。表 3 给出了从 4 种 RNN 学得的 FSA 在分类准确率达到 0.7 时的聚类个数。我们可以发现在五次实验中 MGU 总是最先达到 0.7 的准确率。平均来说，MGU 在聚类个数为 13.6 时准确率达到 0.7，SRN 在聚类个数为 72.2 时准确率达到 0.7，GRU 在聚类个数为 159.4 时准确率达到 0.7，LSTM 在聚类个数设置到 31.8 时准确率才达到 0.7。

图 4 显示 MGU 生成的 FSA 在任务“0110”和任务“000”上随着聚类个数的增加，准确率（分别以 RNN 输出的标记和真实标记作为衡量标准）的变化趋势。从图 4(b)我们可以看出 FSA 和 RNN 输出的标记的一致程度比 FSA 和真实标记（groundtruth，简称 GRD）的一致程度要高，表明 FSA 在一定程度上能够代表 RNN 或者换句话说 FSA 与 RNN 具有统一性。从图 4(b)中还能看出 FSA 的准确率随着聚类个数的增加而不断接近 RNN 的准确率，验证了本节第一段所说的“当对隐层节点聚类的类簇个数 k 足够大的时候，FSA 的性能就应该和 RNN 的性能相近”的论断。在图 4(a)中，FSA 的性能并没有随着聚类个数的增大而增大，并且红色虚线和蓝色实线重合了。这是因为任务“0110”较为简单，FSA 在聚类个数较少的时候就达到了 RNN 的性能，无法继续上升了，所以最终只能在 RNN 的准确率附近波动；而红色虚线和蓝色实线重合是因为 RNN 输出的标记和真实标记相同。

我们再来分析真实任务上的情况。真实任务比模拟任务要困难很多，聚类个数的上限是所有隐层节点输出的个数，而在海量的单词面前，隐层节点的输出也是海量的。所以在真实任务上我们没有办法遍历所有的聚类个数。我们转而去比较最少的情况，也就是聚类个数为 2 的情况，在相同聚类个数的情况下来比较性能，在相同情况下分类准确率越高代表学得的 FSA 性能越好。这种情况下，由于聚类个数少，我们也可以清晰地把 FSA 画出来，从而有利于去分析 RNN 的可解释性。我们分别用 word2vec、GloVe50、GloVe100、GloVe200、GloVe300 对影评里的每一个单词做映射，将得到的向量序列输入到四种 RNN 中做训练和测试，并用从 RNN 中学得的 FSA 对测试集做测试。我们分别记录了五次结果的平均值及其标准差。最后的准确率结果和排序都放到了表 4 中，FSA 比对真实标记的准确率都放在了每种特征对应三行数据的第一行，并用灰色底纹做了标注。从灰色底纹数据中我们可以发现 MGU 基本上在所有的特征下都表现的最好。我们还做了统计显著性检验——t 检验，表 4 中带有黑点的数据表明 MGU 是比其显著好的（分布的均值以 95%的置信度显著好）。可以看出 MGU 比不带门机制的 SRN 要显著好，而并没有显著好于同样带

表 4 聚类个数为 2 时 FSA 在情感分析任务上的准确率及其标准差。灰色底纹数据中加粗的表示的是每一行四种 RNN 中最好的数据，黑点表明 MGU 显著好于被加黑点的数据。

特征	情形	MGU	SRN	GRU	LSTM	均值	排名
Word2vec	比对真实标记	**0.705±.0518**	0.546±.0647•	0.684±.0288	0.679±.0571	0.6535±.0506	3
	比对RNN标记	0.773±.0454	0.725±.1190	0.744±.0581	0.749±.0723	0.7477±.0737	
	RNN 准确率	0.818±.0144	0.599±.0439	0.804±.0329	0.770±.0177	0.7478±.0272	
GloVe50	比对真实标记	**0.732±.0372**	0.581±.0290•	0.713±.0676	0.604±.0694•	0.6575±.0508	**1**
	比对RNN标记	0.809±.0233	0.782±.0344	0.760±.0704	0.676±.0585	0.7568±.0466	
	RNN 准确率	0.823±.0160	0.603±.0275	0.817±.0179	0.786±.0222	0.7573±.0209	
GloVe100	比对真实标记	**0.664±.0711**	0.576±.0152•	**0.664±.0423**	0.636±.0754	0.6350±.0510	5
	比对RNN标记	0.693±.1024	0.742±.0498	0.744±.0397	0.707±.0441	0.7215±.0590	
	RNN 准确率	0.831±.0102	0.634±.0416	0.816±.0198	0.777±.0560	0.7645±.0319	
GloVe200	比对真实标记	**0.699±.0246**	0.591±.0444•	0.664±.0309	0.662±.0437	0.6540±.0359	2
	比对RNN标记	0.745±.0486	0.757±.0454	0.723±.0564	0.742±.0076	0.7418±.0395	
	RNN 准确率	0.820±.0122	0.618±.0236	0.817±.0160	0.792±.0254	0.7618±.0193	
GloVe300	比对真实标记	**0.658±.1085**	0.601±.0139	0.638±.0303	0.652±.0413	0.6373±.0485	4
	比对RNN标记	0.726±.0964	0.760±.0524	0.722±.0228	0.715±.0454	0.7308±.0543	
	RNN 准确率	0.810±.0146	0.641±.0222	0.806±.0270	0.791±.0293	0.762±.0233	

有门机制的 GRU 和 LSTM。其实我们并不期待 MGU 要比它们显著好，毕竟 MGU 的结构在带有门机制的 RNN 里是最简单的，能和其他带有门机制的 RNN 可比就说明其结构设计的优势了。为了表明学得的 FSA 和其对应的 RNN 具有一致性或者 FSA 和 RNN 之间存在强相关性，我们还测试了 FSA 比对 RNN 的输出标记的准确率，放在了表 4 每种特征对应三行数据的第二行。可以看到以 RNN 的输出作为比对标记得到的 FSA 的准确率要比以真实标记作为比对标记得到的 FSA 的准确率要高，这说明在真实任务中 FSA 和 RNN 也具有统一性。我们还在每种特征对应三行数据的第三行提供了 RNN 的准确率，可以看出 RNN 的准确率并没有比 FSA 的准确率高特别多。FSA 的准确率比 RNN 的准确率低是因为为了简化 FSA 的构造，丢弃了一些低频转换，从而丢失了信息。这一问题可以通过不丢弃低频转换或者在相同状态间高频和低频跳转上引入概率或者权重来得到改善。这可以作为后续的工作继续研究，但可能会对可解释性造成影响。所以权衡好可解释性和性能也是一大挑战。

另外，特征维度的升高对性能的提升并没有明显的影响，从表 4 中可以看出，GloVe50 虽然仅有 50 维，但通过其学得的 FSA 的性能表现并不差，甚至可以达到最优。从准确率的绝对数值上来说，无论维度是多少，性能的差别并不大。而较少的维度可以使得 RNN 的神经节点数变少，从而减少空间和时间的消耗。这表明我们为了减少时间和空间复杂度而应该选择较短的特征。这也验证了奥卡姆剃刀原则，也即切勿浪费较多东西，去做“用较少的东西，同样可以做好的事情”。前文中提到的“门很重要，但是越少越好”的结论也验证了这一原则。

4.3 FSA的可视化

为了能够直观清晰地理解 RNN 的内部机制，我们可视化了从 RNN 学得的 FSA。由于从 MGU 学得的 FSA 最简洁，我们从五次试验中随机挑选一个从 MGU 学得的 FSA，并使用 Graphviz[36]工具来将其画出。灰色的圆圈表示开始状态，双圈表示接受状态。也即如果跳转结束，停在双圈内，FSA 输出正类，否则输出负类。

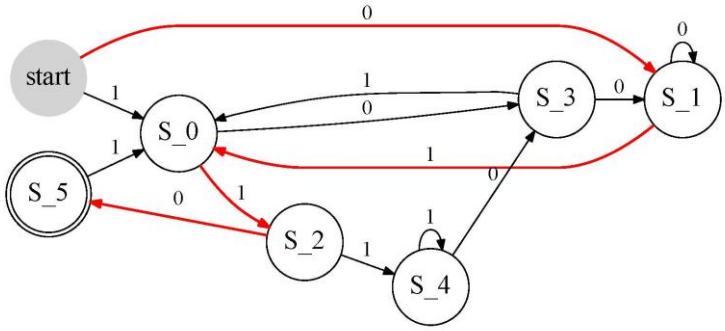


图 5 MGU 在任务“0110”上的 FSA 图。

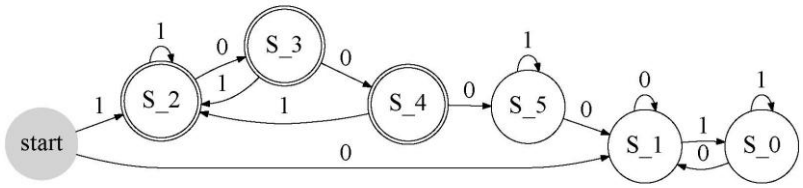


图 6 MGU 在任务“000”上的 FSA 图。

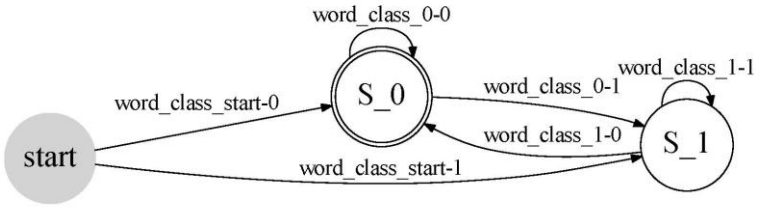


图 7 MGU 在情感分析任务上的 FSA 图。

在任务“0110”中，我们把序列 0110 会经过的路线标成了红色。从图 5 中可以看到，对于所有长度为 4 的 0/1 序列，只有 0110 会沿着红色的路线最终跳转到接受状态，而其他的序列并不能够做到。在这里我们想强调的是，状态的跳转完全是由输入和当前状态决定的，并没有任何的数值计算。所以这种图示满足可解释性的定义，也即是可以被人类模拟的，从而使得人们更容易理解模型运行整个过程发生了什么。

任务“000”比任务“0110”更为复杂。为了可模拟性，我们挑选了和五次实验中聚类个数最少的 FSA 来画图，如图 6 所示。由于这个 FSA 的准确率并不是百分之百，并且接受路径也有多种，所以不再像任务“0110”那样标出某一固定路线。

情感分析的任务里的词汇量要比“0110”任务中仅有的两个要多的多。这也意味着 FSA 里的 Σ 集合里的元素个数要远比 2 大。为了画出一个清晰简洁的图像，我们聚合了两个状态间所有相同方向的边，并只画出了类簇个数为 2 的从 MGU 学得的 FSA 的图像，如图 7 所示。这种情况下，聚合起来的边上的单词会自然地被分成一类，这一类单词在图中被表示为“word_class”。

4.4 可解释性的讨论

在章节 4.3 中，我们给出了从 RNN 中学得的 FSA 的图示，从图示中我们可以清晰地看到 FSA 内部在给定输入后是怎样跳转的，这一点已经从可解释方面给出了一定的启示。而在章节 4.2 中我们发现从 MGU 学得的 FSA 可以在达到较高的准确率的

表 5 引导跳转从 S_1 到接受状态 S_0 的词汇 word_class_1-0。

正面词汇	riffs Wonderful gratitude diligent spectacular sweetness exceptional Best feats sexy bravery beautifully immediacy meditative captures incredible virtues excellent shone honor pleasantly lovingly exhilarating devotion teaming humanity graceful tribute peaking insightful frenetic romping proudly terrific Haunting sophisticated strives exemplary favorite professionalism enjoyable alluring entertaining sorrowful Truly noble bravest exciting Hurray wonderful Miracle
负面词汇	downbeat wicked jailed exceptionally corruption

表 6 引导跳转到从 S_0 到非接受状态 S_1 的词汇 word_class_0-1。

负面词汇	shut dullest unattractive Nothing adulterous stinkers drunken hurt rigid unable confusing risky mediocre nonexistent idles horrible disobeys bother scoff interminably arrogance mislead filthy dependent MISSED asleep unfortunate criticized weary corrupt jeopardized drivel scraps phony prohibited foolish reluctant Ironically fell escape
正面词汇	merry advance excused

同时，聚类个数可以很少，或者在相同聚类个数时，性能更优，这使得 MGU 对应的 FSA 可以有更清晰的图示和内部构造，从而人们更容易理解。考虑到 MGU 比 GRU 和 LSTM 包含更少的门，并且 SRN 没有门，我们认为门是一种正则化，用来控制学得的 FSA 的复杂度，也同样控制隐层节点输出空间的复杂度，而没有门的话则会导致欠拟合。这也给我们一些启示，门机制在 RNN 中是重要的，但门应该越少越好，这为我们设计别的 RNN 可以做出一些指导。具体来说 RNN 其实有很多的变种，比如带注意力机制的 RNN[37]，双向 RNN[38]，巢穴状 RNN13[39]等。这些 RNN 的底层都需要基本的 RNN，那么根据我们的结论，可以在设计这些 RNN 的时候，把底层最基本的 RNN 都换成只有一个门的 RNN，比如 MGU，从而设计出更好的 RNN 模型。

而对于情感分析的任务，我们尝试去寻找其语义上的可解释性。我们在 4.3 节提到由于我们把两个状态间相同方向的跳转集合成了一个，所以引起这些跳转的单词自然地组成一个类。那么我们便期待这一个类里的单词的情感都是一致的。所以我们分析了图 7 中从状态 S_1 到状态 S_0 和从状态 S_0 到状态 S_1 的词类（分别是 word_class_1-0 和 word_class_0-1），并把他们按正面和负面词汇进行了分类，如表 5 和表 6 所示。在剔除小部分的中性词汇后，我们发现 word_class_1-0 里大部分都是正面词汇，只有极少的负面词汇。而 word_class_0-1 大部分都是负面词汇，只有极少数是正面词汇。这很符合我们的预期，因为 S_0 是接受状态，而 S_1 是非接受状态，所以引导跳转到接受状态的词更应该是正面词汇，因为接受状态代表正面，相同地，引导跳转到非接受状态的词应该是负面词汇，因为非接受状态代表负面。这从一定程度上展现了语义上的可解释性以及 RNN 是如何处理富含语义信息的文本类数据的。

5 结束语

随着 RNN 模型变得越来越复杂，而数据的特征也随着环境的动态变化而不断变化，我们很需要在特征变化的情况下去研究 RNN 的可解释性。本文利用从 RNN 中学得的 FSA 来研究 RNN 的可解释性。为了从 RNN 中学得 FSA，我们首先验证了带有门控机制的 RNN 的隐层节点输出也具有聚类特质。通过对隐层节点的输出进行聚类，并把这些类簇当作 FSA 里的状态，我们设计出 FSA 的五个元素。在人工数据集和真实数据集上的分类结果表明具有简单门控结构的 RNN（例如最小门控单元（MGU））是更理想的 RNN 结构。通过图示我们可以人工模拟其运行过程从而清晰地理解其内部运行机制，并且 FSA 中导致特定分类结果的跳转的词具有相同词性，这在语义层面是更容易让人理解的。此外，我们发现不同的特征不会对性能产生严重影响，这表明我们为了减少时间和空间复杂度而应该选择较短的特征。

除了以上贡献外，本文还想传达一个可解释性

的研究思路：我们应该去寻找或者创建和我们要研究的复杂的模型相似且能够代表它的简单模型，并用这种具有代表性的简单模型来研究复杂模型的可解释性。比如本文发现具有简单结构的 FSA 和 RNN 相似，并且实验验证了 FSA 能够一定程度上代表 RNN，从而使用 FSA 来研究 RNN 的可解释性。不过本文工作还有很多不足。比如 FSA 只能做二分类，那么它能够处理的带有序列数据集的任务就只有情感分析任务或和情感分析相似的二分类任务。因此去寻找或者创建能够处理更复杂任务的简单模型将成为未来一个很有趣且重要的研究内容。

致 谢 本文受到国家自然科学基金项目（资源受限的多视图学习技术研究，项目号 61673201）和新型软件技术与产业化协同创新中心的资助。

参 考 文 献

[1] Elman J L. Finding structure in time. Cognitive Science, 1990, 14(2): 179–211

[2] Zhou Z-H. Learnware: on the future of machine learning. Frontiers of Computer Science, 2016, 10(4): 589–590

[3] Kim B, Koyejo O, Khanna R. Examples are not enough, learn to criticize! Criticism for interpretability//Proceedings of the Advances in Neural Information Processing Systems 29. Barcelona, Spain, 2016: 2280–2288

[4] Miller T. Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence, 2019, 267: 1–38

[5] Lipton Z C. The mythos of model interpretability. arXiv preprint arXiv: 1606.03490, 2016

[6] Hou B-J, Zhang L, Zhou Z-H. Learning with feature evolvable streams//Proceedings of the Advances in Neural Information Processing Systems 30. Long Beach, USA, 2017: 1416–1426

[7] Zhou G, Wu J, Zhang C, Zhou Z-H. Minimal gated unit for recurrent neural networks. arXiv preprint arXiv: 1603.09420, 2016.

[8] Cho K, van Merrienboer B, G¨ulc¸ehre C¸, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Doha, Qatar, 2014: 1724–1734

[9] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735–1780

[10] Hinton G, Deng L, Yu D, Dahl G E, Mohamed A-r, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath T N, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 2012, 29: 82–97

[11] Vinyals O, Toshev A, Bengio S, Erhan D. Show and tell: A neural image caption generator//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston, USA, 2015: 3156–3164

[12] Tang D, Qin B, Liu T. Document modeling with gated recurrent neural network for sentiment classification//Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal, 2015:1422–1432

[13] Gill A. Introduction to the theory of finite-state machines. New York: McGraw-Hill, 1962

[14] Gold E M. Complexity of automaton identification from given data. Information and Control, 1978, 37(3): 302–320

[15] Angluin D, Smith C H. Inductive inference: Theory and methods. ACM Computing Surveys, 1983, 15(3): 237–269

[16] Omlin C W, Giles C L. Extraction of rules from discrete-time recurrent neural networks. Neural Networks, 1996, 9(1): 41–52

[17] Hou B-J, Zhou Z-H. Learning with interpretable structure from RNN. arXiv preprint arXiv: 1810.10708, 2018

[18] Arthur D, Vassilvitskii S. k-means++: the advantages of careful seeding//Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. New Orleans, USA, 2007: 1027–1035

[19] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks//Proceedings of the Advances in Neural Information Processing Systems 25. Lake Tahoe, USA, 2012: 1106–1114

[20] Goodfellow I J, Bengio Y, Courville A C. Deep Learning. Cambridge: MIT Press, 2016

[21] Bengio Y, Simard P Y, Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 1994, 5(2): 157–166

[22] Goudreau M W, Giles C L, Chakradhar S T, Chen D. Firstorder versus second-order single-layer recurrent neural networks. IEEE Transactions on Neural Networks, 1994, 5(3): 511–513

[23] Karpathy A, Johnson J, Li F. Visualizing and understanding recurrent networks. arXiv preprint arXiv: 1506.02078, 2015

[24] van der Maaten L. Learning a parametric embedding by preserving local structure// Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics. Clearwater Beach, USA, 2009: 384–391

[25] Yosinski J, Clune J, Nguyen A M, Fuchs T J, Lipson H.

Understanding neural networks through deep visualization. arXiv preprint arXiv: 1506.06579, 2015

[26] Zhou B, Khosla A, Lapedriza A`, Oliva A, Torralba A. Object detectors emerge in deep scene cnns. arXiv preprint arXiv: 1412.6856, 2014

[27] Zeiler M D, Fergus R. Visualizing and understanding convolutional networks//Proceedings in the 13th European Conference on Computer Vision. Zurich Switzerland, 2014: 818–833

[28] Harley A W. An interactive node-link visualization of convolutional neural networks//Proceedings of the 11th International Symposium on Visual Computing. Las Vegas, USA, 2015: 867–877

[29] Rauber P E, Fadel S G, Falc˜ao A X, Telea A C. Visualizing the hidden activity of artificial neural networks. IEEE Transactions on Visualization and Computer Graphics. 2017, 23(1): 101–110

[30] Liu M, Shi J, Li Z, Li C, Zhu J, Liu S. Towards better analysis of deep convolutional neural networks. IEEE Transactions on Visualization and Computer Graphics. 2017, 23(1): 91–100

[31] Zeng Z, Goodman R M, Smyth P. Learning finite state machines with self-clustering recurrent networks. Neural Computation, 1993, 5(6): 976–990

[32] Maas A L, Daly R E, Pham P T, Huang D, Ng A Y, Potts C. Learning word vectors for sentiment analysis//Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics. Portland, USA, 2011: 142–150

[33] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv: 1301.3781, 2013

[34] Pennington J, Socher R, Manning C D. Glove: Global vectors for word representation//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Doha, Qatar, 2014: 1532–1543

[35] Tomita M. Learning of construction of finite automata from examples using hill climbing. Pittsburgh: Carnegie Mellon University, Technical Report: CMU-CS-82-127, 1982

[36] Ellson J, Gansner E R, Koutsofios E, North S C, Woodhull G. Graphviz and dynagraph - static and dynamic graph drawing tools// Jünger M, Mutzel P. Graph Drawing Software. New York: Springer, 2004: 127–148

[37] Qin Y, Song D, Chen H, Cheng W, Jiang G, Cottrell G. A dual-stage attention-based recurrent neural network for time series prediction. arXiv preprint arXiv: 1704.02971, 2017

[38] Jagannatha A N, Yu H. Bidirectional RNN for medical event detection in electronic health records//Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics. San Diego, USA, 2016: 473-482

[39] Moniz J R A, Krueger D. Nested lstms. arXiv preprint arXiv: 1801.10308, 2018

HOU Bo-Jian, born in 1992, Ph. D. candidate. His research interest includes artificial intelligence, machine learning and data mining.

JIANG Yuan, born in 1976, Ph.D., professor, Ph. D. supervisor. His research interest includes artificial intelligence, machine learning and data mining.

Background

Recurrent Neural Networks (RNNs) have been successfully applied to many practical tasks, such as speech recognition, text classification, etc. However, RNNs still have several deficiencies. For example, they lack interpretability like a black boxes, which means their inner working mechanisms are not clearly understood whereas people usually want to know what have been learned by them, particularly in real tasks where decision reliability is crucial and rigorous judgement by human beings are critical. Besides, once they have been trained, if environment changes, which often happens in real tasks, they can hardly perform well or even become useless. These deficiencies have been describe in Learnware (2016) proposed by Prof. Zhi-Hua Zhou from Nanjing University. Considering these issues, Learnware (2016) states that a good model like learnware should be interpretable and evolvable. We follow these two concepts and attempts to explore the interpretability and evolvability of RNNs. Simultaneously studying the interpretability and evolvability of RNNs has not been studied before.

In this paper, for interpretability, we find that Finite State

Automaton (FSA) that processes sequential data has more interpretable inner mechanism and can be learned from RNNs. Besides, in open environment, the features often change and thus for evolvability, we focus on feature changing.

Specifically, we propose a method to learn FSA from RNN and study the impact of feature changing. From the FSA's point of view, we analyze how the performance of RNNs are affected by feature changing and the number of gates. We then give the graphical illustration of FSA for human beings to simulate, which shows the interpretability. With the graphical illustration, we also analyze the semantic meaning behind the transition of numerical hidden states. Our results suggest that RNNs with simple gated structure such as Minimal Gated Unit (MGU) is more desirable and the words on the transitions in FSA leading to specific classification result are within the same class, which are understandable by human beings semantically. Furthermore, we find that different features will not make a serious difference to the performance, which suggest us to choose a short one for the sake of saving time and space complexity.

This work is supported by the National Natural Science Foundation of China under Grant No. 61673201 and the Collaborative Innovation Center of Novel Software Technology and Industrialization.