Implicit N-grams Induced by Recurrence

Xiaobing Sun and **Wei Lu**
StatNLP Research Group
Singapore University of Technology and Design
xiaobing_sun@mymail.sutd.edu.sg, luwei@sutd.edu.sg

Abstract

Although self-attention based models such as Transformers have achieved remarkable successes on natural language processing (NLP) tasks, recent studies reveal that they have limitations on modeling sequential transformations (Hahn, 2020), which may prompt re-examinations of recurrent neural networks (RNNs) that demonstrated impressive results on handling sequential data. Despite many prior attempts to interpret RNNs, their internal mechanisms have not been fully understood, and the question on how exactly they capture sequential features remains largely unclear. In this work, we present a study that shows there actually exist some explainable components that reside within the hidden states, which are reminiscent of the classical n-grams features. We evaluated such extracted explainable features from trained RNNs on downstream sentiment analysis tasks and found they could be used to model interesting linguistic phenomena such as negation and intensification. Furthermore, we examined the efficacy of using such n-gram components alone as encoders on tasks such as sentiment analysis and language modeling, revealing they could be playing important roles in contributing to the overall performance of RNNs. We hope our findings could add interpretability to RNN architectures, and also provide inspirations for proposing new architectures for sequential data.

1 Introduction

Modern recurrent neural networks (RNNs), including Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014), have demonstrated impressive results on tasks involving sequential data. They have proven to be capable of modeling formal languages (Weiss et al., 2018; Merrill, 2019; Merrill et al., 2020) and capturing structural features (Li et al., 2015a,b, 2016; Linzen et al., 2016; Belinkov et al., 2017; Liu et al., 2019) on NLP tasks. Although Transformers (Vaswani et al., 2017) have achieved remarkable performances on NLP tasks such as machine translation, it is argued that they may have limitations on modeling hierarchical structure (Tran et al., 2018; Hahn, 2020) and cannot handle functions requiring sequential processing of input well (Dehghani et al., 2019; Hao et al., 2019; Bhattamishra et al., 2020; Yao et al., 2021). Furthermore, a recent work shows that combining recurrence and attention (Lei, 2021) can result in strong modeling capacity. Another recent work incorporating recurrent cells into Transformers (Hutchins et al., 2022) substantially improved performance on language modeling involving very long sequences, prompting re-investigations of RNNs. On the other hand, it was observed in prior work that RNNs were able to capture linguistic phenomena such as negation and intensification (Li et al., 2016), but the question why they could achieve so still largely remains unanswered.

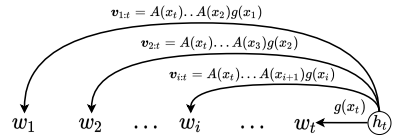


Figure 1: An RNN hidden state may encode a linear combination of all the n-grams ending at the current time step.

In this work, we focus on better understanding RNNs from a more theoretical perspective. We demonstrate that the recurrence mechanism of RNNs may induce a linear combination of interpretable components. These components reside in their hidden states in the form of the iterated matrix-vector multiplication that is based on the representations of tokens in the (reverse) order they appear in the sequence. Such components, solely depending on inputs and learned parameters, can be conveniently interpreted and are reminiscent of those compositional features used in classical n-gram models (Jurafsky and Martin, 2009). They may also provide us with insights on how RNNs compose semantics from basic linguistic units. Our analysis further shows that, the hidden state at each time step includes a weighted combination of components that represent all the "n-grams" ending at that specific position in the sequence as shown

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1624 - 1639

in Figure 1. We gave specific representations for the n-gram components in Elman RNNs (Elman, 1990), GRUs and LSTMs.

We investigated the interpretability of those n-gram components on trained RNN models, and found they could explain phenomena such as negation and intensification and reflect the overall polarity on downstream sentiment analysis tasks, where such linguistic phenomena are prevalent. Our experiments also revealed that the GRU and LSTM models are able to yield better capabilities in modeling such linguistic phenomena than the Elman RNN model, partly attributed to the gating mechanisms they employed which resulted in more expressive n-gram components. We further show that the linear combination of such components yields effective context representations. We explored the effectiveness of such n-gram components (along with the corresponding context representations) as alternatives to standard RNNs, and found they can generally yield better results than the baseline compositional methods on several tasks, including sentiment analysis, relation classification, named entity recognition, and language modeling.

We hope that our work could give inspirations to our community, serving as a useful step towards proposing new architectures for capturing contextual information within sequences.[1]

2 Related Work

Interpretability of RNNs: A line of work focuses on the relationship between RNNs and finite-state machines (Weiss et al., 2018; Merrill, 2019; Suzgun et al., 2019; Merrill et al., 2020; Eyraud and Ayache, 2020; Rabusseau et al., 2019), providing explanation and prediction on the expressive power and limitations of RNNs on formal languages both empirically and theoretically. Kanai et al. (2017) investigated conditions that could prevent gradient explosions for GRU based on dynamics. Maheswaranathan et al. (2019) and Maheswaranathan and Sussillo (2020) linearized the dynamics of RNNs around fixed points of hidden states and elucidated contextual processing. Our work focuses on studying a possible mechanism of RNNs that handles exact linguistic features.

Another line of work aims to detect linguistic features captured by RNNs. Visualization approaches (Karpathy et al., 2015; Li et al., 2016) were initially used to examine compositional information in RNN outputs. Linzen et al. (2016) assessed LSTMs' ability to learn syntactic structure and Emami et al. (2021) gave rigorous explanations on the standard RNNs' ability to capture long-range dependencies. Decomposition methods (Murdoch and Szlam, 2017; Murdoch et al., 2018; Singh et al., 2019; Arras et al., 2017, 2019; Chen et al., 2020) were proposed to produce importance scores for hierarchical interactions in RNN outputs. Our work can be viewed as an investigation on how those interaction came about.

Compositional Models: A variety of compositional functions based on vector spaces have been proposed in the literature to compose semantic meanings of phrases, including simple compositions of adjective-noun phrases represented as matrix-vector multiplication (Mitchell and Lapata, 2008; Baroni and Zamparelli, 2010) and a matrix-space model (Rudolph and Giesbrecht, 2010; Yessenalina and Cardie, 2011) based on matrix multiplication. Socher et al. (2012, 2013) introduced a recursive neural network model that assigns every word and longer phrase in a parse tree both a vector and a matrix, and represents composition of a non-terminal node with matrix-vector multiplication. Kalchbrenner and Blunsom (2013) employed convolutional and recurrent neural networks to model compositionality at the sentence and discourse levels respectively. Those models are designed in an intuitive manner based on the nature of languages thus being interpretable. We can show that RNNs may process contextual information in a way bearing a resemblance to those early models.

3 A Theory on N-gram Representation

First, let us spend some time to discuss how to represent n-grams. Various approaches to representing n-grams have been proposed in the literature (Mitchell and Lapata, 2008; Bengio et al., 2003; Mitchell and Lapata, 2008; Mnih and Teh, 2012; Ganguli et al., 2008; Orhan and Pitkow, 2020; Emami et al., 2021; Rudolph and Giesbrecht, 2010; Yessenalina and Cardie, 2011; Baroni and Zamparelli, 2010). We summarize in Table 1 different approaches for representing n-grams.

Although empirically it has been shown that different approaches can lead to different levels of effectiveness, the rationales underlying many of the design choices remain unclear. In this section, we establish a small theory on representing n-grams, which leads to a new formulation on capturing the semantic information within n-grams.

Let us assume we have a vocabulary $\mathbb{V}$ that consists of all possible word tokens. The set of n-grams can be denoted as $\mathbb{V}^*$ (including the special

[1] Our code is available at https://github.com/richardsun-voyager/inibr.

Model	N-gram Representation	Context Representation	L	Representative Work
Vector Multiplicative (**VM**)	$\boldsymbol{v}_{i:j} = g(x_i) \odot \cdots \odot g(x_j)$	$\boldsymbol{v}_{1:t}$	t	Mitchell and Lapata (2008)
Matrix Multiplicative (**MM**)	$\boldsymbol{M}_{i:j} = \prod_{k=i}^{j} A(x_k)$	$\boldsymbol{M}_{1:t}$	t	Yessenalina and Cardie (2011)
Vector Additive (*weighted*) (**VA-W**)	$\boldsymbol{v}_{i:j} = \boldsymbol{C}_{j-i} g(x_i)$	$\sum_{i=t-m+1}^{t} \boldsymbol{v}_{i:t}$	m	Bengio et al. (2003)
Vector Additive (*exponentially weighted*) (**VA-EW**)	$\boldsymbol{v}_{i:j} = \boldsymbol{C}^{j-i} g(x_i)$	$\sum_{i=1}^{t} \boldsymbol{v}_{i:t}$	t	Emami et al. (2021)
Matrix-Vector Multiplicative (*restricted*) (**MVM-R**)	$\boldsymbol{v}_{i-1:i} = A(x_{i-1}) g(x_i)$	$\boldsymbol{v}_{t-1:t}$	2	Baroni and Zamparelli (2010)
Matrix-Vector Multiplicative (**MVM**)	$\boldsymbol{v}_{i:j} = \left(\prod_{k=j}^{i+1} A(x_k)\right) g(x_i)$	$\boldsymbol{v}_{1:t}$	t	-
Matrix-Vector Multiplicative-Additive (**MVMA**)	$\boldsymbol{v}_{i:j} = \left(\prod_{k=j}^{i+1} A(x_k)\right) g(x_i)$	$\sum_{i=1}^{t} \boldsymbol{v}_{i:t}$	t	This work

Table 1: Different models for defining representations for n-grams within the phrase $x_1, x_2, \ldots, x_{t-1}, x_t$ and constructing the context representation out of the n-grams during learning. L: the maximum length allowed for the context representation. $\boldsymbol{C}$ is a weight matrix, and $\boldsymbol{C}_k$ is a (relative) position-specific weight matrix. A and g are functions that return a matrix and a vector respectively.

n-gram which is the empty string ϵ). Consider three n-grams a, b, and c from $\mathbb{V}^*$, with their semantic representations $r(a)$, $r(b)$, and $r(c)$ respectively. Similarly, we may have $r(ab)$ which return the semantic representations of the concatenated n-grams ab. It is desirable for our representations to be compositional in some sense. Specifically, a longer n-gram may be semantically related to those shorter n-grams it contains in some way.

Under some mild compositional assumptions related to the *principle of compositionality* (Frege, 1948)[2], it is reasonable to expect that there exists some sort of rule or operation that allows us to compose semantics of longer n-grams out of shorter ones. Let us use $\otimes$ to denote such an operation. We believe a good representation system for n-grams shall satisfy several key properties. First, the semantics of the n-gram abc shall be determined through either combining the semantics of the two n-grams a and bc or through combining the semantics of ab and c. The semantics of abc is unique, regardless of which of these two ways we use. Second, for the empty string ϵ, it should not convey any semantics. Formally, we can write them as:[3]

- **Associativity**: $\forall a, b, c \in \mathbb{V}^*$, $(r(a) \otimes r(b)) \otimes r(c) = r(a) \otimes (r(b) \otimes r(c))$
- **Identity**: $\forall a \in \mathbb{V}^*$, $r(a) \otimes r(\epsilon) = r(a)$, and $r(\epsilon) \otimes r(a) = r(a)$

This essentially shows that the representation space for all n-grams under the operation $\otimes$, denoted as $(\mathbb{V}^*, \otimes)$, forms a *monoid*, an important concept in abstract algebra (Lallement, 1979), with significance in theoretical computer science (Meseguer and Montanari, 1990; Rozenberg and Salomaa, 2012).

On the other hand, it can be easily verified that the space of all $d \times d$ (where d is an integer) real square matrices under matrix multiplication, denoted as $(\mathbb{R}^{d \times d}, \cdot)$, also strictly forms a monoid (i.e., it is associative and has an identity, but is *not* commutative). We can therefore establish a homomorphism from $\mathbb{V}^*$ to $\mathbb{R}^{d \times d}$, resulting in the function $r(\cdot) \in \mathbb{V}^* \rightarrow \mathbb{R}^{d \times d}$.

This essentially means that we may be able to rely on a sub-space within $\mathbb{R}^{d \times d}$ as our mathematical object to represent the space of n-grams, where the matrix multiplication operation can be used to compose representations for longer n-grams from shorter ones. Thus, for a unigram x (a single word in the vocabulary), we have:

$$r(x) := \boldsymbol{A}_x \tag{1}$$

where $\boldsymbol{A}_x \in \mathbb{R}^{d \times d}$ is the representation for the word x (how to learn such a matrix is a separate question to be discussed later). Note that the empty string ϵ comes with a unique representation which is the $d \times d$ identity matrix $\boldsymbol{I}$.

We can either use matrix left-multiplication or right-multiplication as our operator $\otimes$. Assume the language under consideration employs the left-to-right writing system. It is reasonable to believe that a human reader processes the text left-to-right, and the semantics of the text gets evolved each time the reader sees a new word. We may use the matrix left-multiplication as the preferred operator in this case. The system will left-multiply (modify) an existing n-gram representation with a matrix associated with the new word that appears right after the existing n-gram, forming the representation of the new n-gram. Such an operation essentially performs a transform that simulates the process of yielding new semantics when appending a new word at the end of an existing phrase. With this, for

[2]The principle states that "the meaning of an expression is determined by the meanings of the sub-expressions it contains and the rules used to combine such sub-expressions".

[3]Besides, another important property is that the order used for combining two n-grams does matter. In other words, $r(a) \otimes r(b)$ usually may not be the same as $r(b) \otimes r(a)$.

a general n-gram $x_i, x_{i+1}, \ldots, x_t$ ($i \leq t$), we have:

$$r(x_i, x_{i+1}, \ldots, x_t) = \prod_{k=t}^{i} \boldsymbol{A}_{x_k} \tag{2}$$

However, the conventional wisdom in NLP has been to use vectors to represent basic linguistic units such as words, phrases or sentences (Mikolov et al., 2013a,b; Pennington et al., 2014; Kiros et al., 2015). This can be achieved by a transform:

$$\left(\prod_{k=t}^{i} \boldsymbol{A}_{x_k}\right) \boldsymbol{u} \tag{3}$$

where $\boldsymbol{u} \in \mathbb{R}^d$ is a vector that maps the resulting matrix representation into a vector representation.

Next, we will embark on our journey to examine the internal representations of RNNs. As we will see, interestingly, our developed n-gram representations can emerge within such models.

4 Interpretable Components in RNNs

An RNN is a parameterized function whose hidden state can be written recursively as:

$$\boldsymbol{h}_t = f(x_t, \boldsymbol{h}_{t-1}), \tag{4}$$

where x_t is the input token at time step t and $\boldsymbol{h}_{t-1} \in \mathbb{R}^d$ is the previous hidden state. Assume f is differentiable at any point, with the Taylor expansion, $\boldsymbol{h}_t$ can be rewritten as:

$$\boldsymbol{h}_t = f(x_t, \boldsymbol{0}) + \nabla f(x_t, \boldsymbol{0})\boldsymbol{h}_{t-1} + o(\boldsymbol{h}_{t-1}), \tag{5}$$

where $\nabla f(x_t, \boldsymbol{0}) = \frac{\partial f}{\partial \boldsymbol{h}_{t-1}}|_{\boldsymbol{h}_{t-1}=\boldsymbol{0}}$ is the Jacobian matrix, and o is the remainder of the Taylor series.

Let $g(x_t) = f(x_t, \boldsymbol{0})$ and $A(x_t) = \nabla f(x_t, \boldsymbol{0})$. Note that $g(x_t) \in \mathbb{R}^d$ and $A(x_t) \in \mathbb{R}^{d\times d}$ are both functions of x_t. Therefore, the equation above can be written as:

$$\boldsymbol{h}_t = g(x_t) + A(x_t)\boldsymbol{h}_{t-1} + o(\boldsymbol{h}_{t-1}). \tag{6}$$

If the hidden state has a sufficiently small norm, it can be approximated by the first-order Taylor expansion as follows[4]:

$$\boldsymbol{h}_t \approx g(x_t) + A(x_t)\boldsymbol{h}_{t-1}. \tag{7}$$

Next we illustrate how this recurrence relation can help us identify some salient components.

[4]There will be an "approximation gap" at each time step between the "approximated" hidden state and the actual standard hidden state. We may leverage regularization methods such as weight-decaying and the spectral normalization (Miyato et al., 2018) to prevent the gap from growing unbounded.

4.1 Emergence of N-grams

Consider the simplified RNN with the following recurrence relation,

$$\boldsymbol{h}_t = g(x_t) + A(x_t)\boldsymbol{h}_{t-1}, \tag{8}$$

where $\boldsymbol{h} \in \mathbb{R}^d$, and $g(x_t) \in \mathbb{R}^d$ and $A(x_t) \in \mathbb{R}^{d\times d}$ are functions of x_t. This recurrence relation can be expanded repeatedly as follows,

$$\begin{aligned}\boldsymbol{h}_t &= g(x_t) + A(x_t)g(x_{t-1}) + A(x_t)A(x_{t-1})\boldsymbol{h}_{t-2} \\ &= \cdots = \sum_{i=1}^{t} A(x_t)\ldots A(x_{i+1})g(x_i) \\ &= \sum_{i=1}^{t} \underbrace{\left(\prod_{j=t}^{i+1} A(x_j)\right) g(x_i)}_{\boldsymbol{v}_{i:t}},\end{aligned}$$

We can see that $\boldsymbol{v}_{i:t}$ bear some resemblance to the term in Equation 3, which can be rewritten as:

$$\left(\prod_{j=t}^{i+1} \underbrace{\boldsymbol{A}_{x_j}}_{A(x_j)}\right) \underbrace{\left(\boldsymbol{A}_{x_i}\boldsymbol{u}\right)}_{g(x_i)}, \tag{9}$$

With the definition $A(x_j) := \boldsymbol{A}_{x_j}$ and $g(x_i) := A(x_i)\boldsymbol{u}$, we can see $\boldsymbol{v}_{i:t}$ can be interpreted as an "n-gram representation" that we developed in the previous section. It is important to note that, however, the use of function $g(x_i)$ in RNNs may lead to greater expressive power than the original formulation based on $\boldsymbol{A}_{x_i}\boldsymbol{u}$.[5]

This interesting result shows that the hidden state of a simple RNN (characterized by Equation 8) is the sum of the representations of all the n-grams ending at time step t. Such salient components within RNN also show that the standard RNN may actually have a mechanism that is able to capture implicit n-gram information as described above. This leads to the following definition:

Definition 1 (N-gram Representation) *For the n-gram $x_i, x_{i+1}, \ldots, x_t$, its representation is:*

$$\boldsymbol{v}_{i:t} = \left(\prod_{j=t}^{i+1} A(x_j)\right) g(x_i), \tag{10}$$

where $A(x_j) \in \mathbb{R}^{d\times d}$ and $g(x_i) \in \mathbb{R}^d$.

4.2 Context Representation

With the above definition, we may want to consider how to perform learning. The learning task involves identifying the functions A and g – in other words, learning representations for word tokens.

A typical learning setup that we may consider here is the task of language modeling. Such a task

[5]This is because we can always construct $g(x_i)$ from any given $\boldsymbol{A}_{x_i}$ and $\boldsymbol{u}$, but in general we may not always be able to decompose $g(x_i)$ into the form $\boldsymbol{A}_{x_i}\boldsymbol{u}$ (for all x_i).

can be defined as predicting the next word x_{t+1} based on the representation of preceding words $x_1, x_2, \ldots, x_t$ which serves as its left context. This is an unsupervised learning task, where the underlying assumption involved is the *distributional hypothesis* (Harris, 1954). Specifically, the model learns how to "reconstruct" the current word x_{t+1} out of $x_1, x_2, \ldots, x_t$ which serves as its context.

Now the research question is how to define the representation for this specific context. As this left context is also an n-gram, it might be tempting to directly use its n-gram representation defined above to characterize such a left context. However, we show such an approach is not desirable.

The n-gram representation for this context can be written in the following alternative form:

$$\boldsymbol{v}_{1:t} = \left(\prod_{j=t}^{2} A(x_j)\right) g(x_i) = W(x_{2:t}) g(x_1), \tag{11}$$

This shows that the n-gram representation of $x_1, x_2, \ldots, x_t$ could be interpreted as a "weighted" representation of the word x_1 (where the weight matrix is derived from the words between x_1 and x_{t+1}, measuring the strength of the connection between them). However, ideally, the context representation shall not just take x_1 but other adjacent words preceding x_{t+1} into account, where each word contributes towards the final context representation based on the connection between them. This leads to the following way of defining the context:

$$\begin{aligned} \sum_{i=1}^{t} \boldsymbol{v}_{i:t} &= \sum_{i=1}^{t} \left(\prod_{j=t}^{i+1} A(x_j)\right) g(x_i) \\ &= \sum_{i=1}^{t} W(x_{i:t}) g(x_i), \end{aligned} \tag{12}$$

In fact, such an idea of defining the context as a weighted combination of surrounding words is not new – it recurs in the literature of language modeling (Bengio et al., 2003; Mnih and Teh, 2012), word embedding learning (Mikolov et al., 2013a,b), and graph representation learning (Cao et al., 2016).

Interestingly, the hidden states in the RNNs, as shown in Equation 9, also suggest exactly the same way of defining this left context. Indeed, when using RNNs for language modeling, each hidden state is exactly serving as the context representation for predicting the next word in the sequence.

The above gives rise to the following definition:

Definition 2 (Context Representation) *For the n-gram $x_1, x_2, \ldots, x_t$, its representation when serving as the (left) context is:*

$$\boldsymbol{c}_{1:t} = \sum_{i=1}^{t} \boldsymbol{v}_{i:t} = \sum_{i=1}^{t} \left(\prod_{j=t}^{i+1} A(x_j)\right) g(x_i), \tag{13}$$

where $A(x_j) \in \mathbb{R}^{d \times d}$ and $g(x_i) \in \mathbb{R}^d$.

4.3 Model Parameterization

With the above understandings on such salient components within RNNs, we can now look into how different variants of RNNs parameterize the functions A and g. The definition of Elman RNN, GRU and LSTM together with the corresponding Jacobian matrix $A(x_t)$ and vector function $g(x_t)$ functions are listed in Table 2[6]. We discuss how such different parameterizations may lead to different expressive power when they are used in practice.

We can see the ways GRU or LSTM parameterize $A(x_t)$ and $g(x_t)$ appear to be more complex compared to Elman RNN. This can partially be attributed to their gating mechanisms. Although the original main motivation of introducing such mechanisms may be to alleviate the exploding gradient and vanishing gradient issues (Hochreiter and Schmidhuber, 1997; Cho et al., 2014), we could see such designs also result in terms describing gates and intermediate representations. A and g are then independently derived based on certain rich interactions between such terms. We believe such interactions may likely increase the expressive power of the resulting n-gram representations. We will validate these points and discuss more in our experiments.

5 Experiments

In our experiments, we focus on the following aspects: 1) understanding the effectiveness of the proposed n-gram (and context) representations when used in practice, as compared to baseline models; 2) examining the significance of the choice of context representation; 3) interpreting the proposed representations by examining how well they could be used to capture certain linguistic phenomena.

We employ the sentiment analysis, relation classification, named entity recognition (NER) and language modeling tasks as testbeds. The first task is often used in investigating n-gram phenomena (Yessenalina and Cardie, 2011; Li et al., 2016) while the others are often used in examining how capable an encoder is when extracting features from texts (Grave et al., 2018; Zhou et al., 2016; Lample et al., 2016).

Datasets For sentiment analysis, we considered the Stanford Sentiment Treebank (SST) (Socher et al., 2013), the IMDB (Maas et al., 2011), and the AG-news topic classification[7] (Zhang et al.,

[6] For brevity, we suppress biases following Merrill et al. (2020).

[7] AG-news can be viewed as a special sentiment analysis dataset.

	Definition	Parameterization	
Elman	$\boldsymbol{h}_t = \tanh(\boldsymbol{W}_{in}\boldsymbol{x}_t + \boldsymbol{W}_{ih}\boldsymbol{h}_{t-1})$	$A(x_t) = \mathrm{diag}[\tanh'(\boldsymbol{W}_{in}\boldsymbol{x}_t)]\boldsymbol{W}_{ih}$	$g(x_t) = \tanh(\boldsymbol{W}_{in}\boldsymbol{x}_t)$.
GRU	$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_{ir}\boldsymbol{x}_t + \boldsymbol{W}_{hr}\boldsymbol{h}_{t-1})$ $\boldsymbol{z}_t = \sigma(\boldsymbol{W}_{iz}\boldsymbol{x}_t + \boldsymbol{W}_{hz}\boldsymbol{h}_{t-1})$ $\boldsymbol{n}_t = \tanh(\boldsymbol{W}_{in}\boldsymbol{x}_t + \boldsymbol{r}_t \odot \boldsymbol{W}_{hn}\boldsymbol{h}_{t-1})$ $\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{n}_t + \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1}$	$A(x_t) = \mathrm{diag}\,[f_n(x_t) \odot [1 - g_z(x_t)]] \odot g_r(x_t)]\boldsymbol{W}_{hn}$ $- \mathrm{diag}[g_n(x_t) \odot f_z(x_t)]\boldsymbol{W}_{hz}$ $+ \mathrm{diag}[g_z(x_t)]$ $g(x_t) = [1 - g_z(x_t)] \odot g_n(x_t)$	where: $g_r(x_t) = \sigma(\boldsymbol{W}_{ir}\boldsymbol{x}_t)$, $f_r(t) = g'_r(x_t)$, $g_z(x_t) = \sigma(\boldsymbol{W}_{iz}\boldsymbol{x}_t)$, $f_z(x_t) = g'_z(x_t)$, $g_n(x_t) = \tanh(\boldsymbol{W}_{in}\boldsymbol{x}_t)$, $f_n(x_t) = g'_n(x_t)$.
LSTM	$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{ii}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1})$ $\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{if}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1})$ $\boldsymbol{o}_t = \sigma(\boldsymbol{W}_{io}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1})$ $\boldsymbol{c}^m_t = \tanh(\boldsymbol{W}_{ic}\boldsymbol{x}_t + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1})$ $\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{c}^m_t$ $\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)$	$A(x_t) = \begin{bmatrix} \boldsymbol{B}_t & \boldsymbol{D}_t \\ \boldsymbol{E}_t & \boldsymbol{F}_t \end{bmatrix}$, $g(x_t) = \begin{bmatrix} g_c(x_t) \\ g_h(x_t) \end{bmatrix}$ $\boldsymbol{B}_t = \mathrm{diag}[g_f(x_t)]$ $\boldsymbol{E}_t = \mathrm{diag}\,[g_o(x_t) \odot \tanh'[g_c(x_t)]]\,\boldsymbol{B}_t$ $\boldsymbol{D}_t = \mathrm{diag}[g^m_c(x_t) \odot f_i(x_t)]\boldsymbol{W}_{hi}$ $+ \mathrm{diag}[g_i(x_t) \odot f^m_c(t)]\boldsymbol{W}_{hc}$ $\boldsymbol{F}_t = \mathrm{diag}\,[g_o(x_t) \odot \tanh'[g_c(x_t)]]\,\boldsymbol{D}_t$ $+ \mathrm{diag}\,[f_o(x_t) \odot \tanh[g_c(x_t)]]\,\boldsymbol{W}_{ho}$	where: $g_c(x_t) = g_i(x_t) \odot g^m_c(x_t)$, $g_h(x_t) = g_o(x_t) \odot \tanh[g_c(x_t)]$, $g_i(x_t) = \sigma(\boldsymbol{W}_{ii}\boldsymbol{x}_t)$, $f_i(x_t) = g'_i(x_t)$, $g_f(x_t) = \sigma(\boldsymbol{W}_{if}\boldsymbol{x}_t)$, $f_f(x_t) = g'_f(x_t)$, $g_o(x_t) = \sigma(\boldsymbol{W}_{io}\boldsymbol{x}_t)$, $f_o(x_t) = g'_o(x_t)$, $g^m_c(x_t) = \tanh(\boldsymbol{W}_{ic}\boldsymbol{x}_t)$, $f^m_c(x_t) = \tanh'(\boldsymbol{W}_{ic}\boldsymbol{x}_t)$.

Table 2: Parameterization of A and g by Elman RNN, GRU, and LSTM. $\boldsymbol{x}_t$ is the representation of the input token x_t and $\boldsymbol{W}_{**}$ refers to a weight matrix. σ and $\tanh$ are the *element-wise* sigmoid and $\tanh$ functions respectively. g', $\tanh'$ and f' refer to the *element-wise* derivative. The diag operation converts a vector into a diagonal matrix.

2015) datasets. The first dataset has sufficient labels for phrase-level analysis, the second dataset has instances with relatively longer lengths, and the third one is multi-class. For relation classification and NER, we considered the SemEval 2010 Task 8 (Hendrickx et al., 2010) and CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003) datasets respectively. For language modeling, we considered the Penn Treebank (PTB) dataset (Marcus et al., 1993), the Wikitext-2 (Wiki2) dataset and the Wikitext-103 (Wiki103) dataset (Merity et al., 2016). PTB is relatively small while Wiki103 is large. The statistics are shown in Tables 6 and 7 in the appendix.

Baselines The n-gram representations (together with their corresponding context representations) discussed in the literature are considered as baselines, which are listed in Table 1 along with the MVMA and MVM models. MVM(A)-G/L/E refers to the MVM(A) model created with the A and g functions derived from GRU/LSTM/Elman, but are trained directly from data. The A and g functions for GRU, LSTM and Elman are listed in Table 2.

Additionally, to understand whether the complexity of A affects the expressive power, we created a new model called MVMA-ME, which comes with an A function that is slightly more complex than that of MVMA-E but less complex than those of MVMA-G and MVMA-L: $A(x_t) = 0.25\,\mathrm{diag}[\tanh(\boldsymbol{W}\boldsymbol{x}_t)]\boldsymbol{M} + 0.5\boldsymbol{I}$ and $g(x_t) = \tanh(\boldsymbol{W}'\boldsymbol{x}_t)$ (here, $\boldsymbol{W}$, $\boldsymbol{M}$ and $\boldsymbol{W}'$ are learnable weight matrices). The g function is the same as that of MVMA-E.

Setup For sentiment analysis, relation classification and language modeling, models consist of one embedding layer, one RNN layer, and one fully-connected layer. The Adagrad optimizer (Duchi et al., 2011) was used along with dropout (Srivastava et al., 2014) for sentiment analysis[8] and relation classification. For language modeling, models were trained with the Adam optimizer (Kingma and Ba, 2014). We ran word-level models with truncated backpropagation through time (Williams and Peng, 1990) where the truncated length was set to 35. Adaptive softmax (Joulin et al., 2017) was used for Wiki103. For NER, models consist of one embedding layer, one bidirectional RNN layer, one projection layer and one conditional random field (CRF) layer. The SGD optimizer was used. Final models were chosen based on the best validation results. More implementation details can be found in the appendix.

Model	SST-2		AG-news		IMDB	
	dev	test	dev	test	dev	test
MM	86.0±1.3	85.6±0.4	-	-	-	-
VA-W	80.6±1.6	80.4±1.4	90.3±0.4	90.0±0.3	88.0±0.6	88.0±0.4
VA-EW	82.6±0.3	82.0±0.3	-	-	-	-
MVM-G	84.9±0.5	85.0±1.0	84.9±4.0	84.4±4.0	50.9±0.0	50.2±0.1
MVM-L	85.4±0.4	84.9±0.8	86.9±1.7	86.5±1.7	51.0±0.1	50.2±0.1
MVM-E	59.6±1.6	59.5±1.1	-	-	-	-
MVMA-G	87.0±0.4	85.3±0.5	91.6±0.5	91.3±0.3	90.5±0.5	89.6±0.7
MVMA-L	86.7±1.0	85.4±1.0	91.4±0.5	91.3±0.5	89.4±0.6	89.2±0.6
MVMA-E	81.4±1.1	80.8±1.5	-	-	-	-
MVMA-ME	83.2±0.5	81.9±0.3	90.6±0.5	90.2±0.3	80.6±0.5	80.1±1.1
GRU	84.9±0.9	84.9±0.5	92.1±0.1	91.6±0.3	87.7±0.2	87.2±0.3
LSTM	84.3±0.8	84.4±0.3	91.9±0.4	91.5±0.5	89.0±0.1	88.7±0.4
Elman	79.1±0.3	79.7±1.4	87.5±0.5	87.5±0.6	67.0±1.9	66.7±0.9

Table 3: Accuracy percentage (↑) on sentiment analysis (text classification) datasets (averaged over 3 runs). "-" means the model failed to converge.

5.1 Comparison of Representation Models

We investigate how baseline n-gram representation models[9], the MVM model, and the MVMA model perform on the aforementioned testbeds. We also compare with the standard RNN models.

Sentiment Analysis Apart from the GRU and LSTM models, it can be observed that our MVMA-G and MVMA-L models are also able to achieve competitive results on three sentiment analysis datasets, as we can see from Table 3, demonstrating the efficacy of those recurrence-induced n-gram

[8] We investigated the approximation between RNNs and their corresponding recurrence relations in Appendix B.2. The spectral normalization (Miyato et al., 2018) was used on the weight matrices $\boldsymbol{W}_{h*}$ for standard RNNs.

[9] We excluded VM, which we found was hard to train. We also excluded MVM-R which only considers bigrams.

representations. Although Elman RNN and its corresponding MVMA-E and MVM-E models also have a mechanism for capturing n-gram information (similar to GRU and LSTM), they did not perform well, which may be attributed to a limited expressive power of their A and g functions when used for defining n-grams as described previously.

Both MM and VA-EW fail to converge on AG-news and IMDB, showing challenges for them to handle long instances. This may be explained by the lengthy matrix multiplication involved in their representations, which may result in vanishing/exploding gradient issues. Interestingly, MVM-G and MVM-L, which solely rely on the longest n-gram representation, are also able to achieve good results on SST-2, indicating a reasonable expressive power of such n-gram representations alone. However, they fail to catch up with MVMA-G and MVMA-L on IMDB which contains much longer instances, confirming the significance of the context representation, which captures n-grams of varying lengths.

Unlike MVMA-E, the MVMA-ME model does not suffer from loss stagnation on AG-news and IMDB but the performance on IMDB obviously falls behind MVMA-G and MVMA-L as shown in Table 3. This indicates a sufficiently expressive $A(x_t)$ (such as the Jacobian matrices of GRU and LSTM) may be needed to handle long instances.

Relation Classification & NER For relation classification, context representations (or final hidden states) are used for classification. For NER, we use the concatenated context representations (or hidden states) at each position of bidirectional models to predict entities and their types. Table 4 shows that MVMA-G and MVMA-L outperform the MVM-G and MVM-L models respectively on both tasks, again confirming the effectiveness of the context representations. MVM(A)-E did not perform as well as MVM(A)-G and MVM(A)-L, which demonstrates the significance of expressive power for the A and g functions. Similar to the results in sentiment analysis, MVMA-ME did not perform as well as MVMA-G and MVMA-L. However, to our surprise, MVMA-ME did not outperform VA-EW on NER, suggesting that a delicate choice of A can be important for this task. The poor performance of VA-W on NER might be explained by a weak expressive power of its n-gram representations. MM fails to converge on the relation classification task, which implies it is not robust across different datasets. Interestingly, it is remarkable that MVMA-G, MVMA-L and MVMA-E could yield competitive results compared to GRU, LSTM and Elman on NER, implying such n-gram representations could be crucial for our NER task.

Model	Relation Classification		NER	
	dev	test	dev	test
MM	-	-	33.9±0.6	30.8±0.4
VA-W	41.2±0.2	37.9±0.9	17.6±0.6	16.5±1.6
VA-EW	39.7±1.1	38.3±0.7	70.8±0.7	63.4±1.0
MVM-G	51.2±0.5	52.6±0.7	54.2±1.6	47.6±2.2
MVM-L	48.8±1.3	50.5±1.5	53.8±1.7	46.6±1.6
MVM-E	-	-	27.8±0.9	25.6±0.9
MVMA-G	62.2±1.0	59.7±0.1	75.0±0.4	67.7±0.5
MVMA-L	57.5±0.3	56.2±0.8	75.6±0.2	67.9±0.3
MVMA-E	27.8±0.9	25.6±0.9	69.0±0.4	61.7±0.1
MVMA-ME	46.3±0.9	46.2±0.6	67.0±0.5	57.6±0.8
GRU	67.2±0.6	62.2±0.2	75.6±0.5	67.9±0.5
LSTM	65.2±0.9	61.3±1.4	76.3±0.5	68.1±0.5
Elman	27.8±0.9	25.6±0.9	67.1±0.9	58.6±0.6

Table 4: F1 scores (↑) (averaged over 3 runs) on the relation classification and NER tasks. "-" means the model failed to converge.

Model		PTB	Wiki2	Wiki103
GRU	dev	118.4±0.4	146.1±0.4	109.4±0.6
	test	110.1±0.4	136.8±0.1	113.3±0.8
MVMA-G	dev	119.8±0.4	150.3±0.8	111.8±0.5
	test	111.1±0.2	140.2±1.0	115.2±0.5
MVM-G	dev	146.5±1.3	170.1±2.8	-
	test	138.8 ±1.0	160.0±2.6	-
LSTM	valid	118.6±0.4	150.6±0.6	108.3±0.6
	test	109.8±0.4	140.4±0.8	112.4±0.8
MVMA-L	dev	121.5±0.5	152.0±0.5	109.1±0.6
	test	113.2±0.5	142.5±0.7	112.6±0.6
MVM-L	dev	124.3±1.5	155.6±0.9	-
	test	117.0±1.0	145.7±1.6	-
MVMA-ME	dev	140.7±0.9	169.0±1.0	153.1±4.2
	test	134.0±1.0	158.4±1.4	157.4±4.3

Table 5: Perplexities (↓) on language modeling (averaged over 5 runs). "-": the model failed to converge.

Language Modeling For the language modeling task, we choose MVMA-G, MVMA-L, MVM-G and MVM-L for experiments. We also run MVMA-ME. As we can see from Table 5, there are performance gaps between the MVMA models and the standard RNNs – though the gaps often do not appear to be particularly large. This indicates there may be extra information within higher order terms of the standard RNN functions useful for such a task. Yet, such information cannot be captured by the MVMA models that employ simplified functions. The gaps between the MVM models and MVMA models are remarkable, which again indicates that the correct way of defining the left context representation can be crucial for the task of next word prediction. MVMA-ME did not perform well on the language modeling task, which might be attributed to the less expressive power of its functions A and g.

5.2 Interpretation Analysis

We conduct some further analysis to examine the interpretability of n-gram representations. Specifically, we examine whether the models are able to capture certain linguistic phenomena such as negation, which is important for sentiment analysis (Ribeiro et al., 2020). We also additionally

made comparisons with the vanilla Transformer (Vaswani et al., 2017) here[10] despite the fact that it remains largely unclear how it precisely captures sequence features such as n-grams.

We could also obtain the n-gram representations and the corresponding context representations from the learned standard RNN models, based on their learned parameters. We denote such n-gram representations as $\text{RNN}_{n\text{-gram}}$, and the context representations as $\text{RNN}_{\text{context}}$, where "RNN" can be GRU, LSTM or Elman. As n-gram representations are vectors, a common approach is to transform them into scalars with learnable parameters (Murdoch et al., 2018; Sun and Lu, 2020). We define the *n-gram polarity score* to quantify the polarity information as captured by an n-gram representation $\boldsymbol{v}_{i:t}$ from time step i to t, which is calculated as:

$$s^{\boldsymbol{v}}_{i:t} = \boldsymbol{w}^\top \boldsymbol{v}_{i:t}, \tag{14}$$

where $\boldsymbol{w}$ is the learnable weight vector of the final fully-connected layer. We also define the *context polarity score* for the context as $\sum_{i=1}^{t} s^{\boldsymbol{v}}_{i:t}$.

We trained RNNs and baseline models on SST-2 and automatically extracted 73 positive adjectives (e.g., "*nice*" and "*enjoyable*") and 47 negative adjectives (e.g., "*bad*" and "*tedious*") from the vocabulary[11]. N-gram polarity scores were calculated for those adjective unigrams and their negation bigrams formed by prepending "*not*" to them. For VA-EW and VA-W, their n-gram representations do not involve tokens other than the last token. Such limitations prevent them from capturing any negation information. We therefore calculate the context polarity scores using their context representations instead (which in this case is a bigram). This also applies to Transformer for the same reason.

We observed that, for the GRU and LSTM models along with their corresponding MVMA models, the n-gram representations are generally able to learn the negation for both the adjective and their negation bigrams as shown in Figures 2a and 2b[12], prepending "*not*" to an adjective will likely reverse the polarity. This might be a reason why they could achieve relatively higher accuracy on the sentiment analysis tasks. Interestingly, MVM-G could also capture negation as shown in Figure 2c, again suggesting the impressive expressive power of such n-gram representations alone.

[10]The mean of output representations was treated as the context representation for Transformer during training. We also tried to use the concatenation of the first and last token, following (Luan et al., 2019), which yielded similar results.

[11]Such adjectives and detailed automatic extraction process can be found in the appendix.

[12]Results of LSTM are similar to GRU, which can be found in the appendix.

However, as shown in Figure 2, models such as VA-W, MVMA-E, and MM are struggling to capture negation for negative adjectives, again implying a weaker expressive power of their n-gram representations. Specifically, MVMA-E fails to capture negation for negative adjectives, which may be attributed to a relatively weaker Jacobian matrix function A (as compared to those of GRU and LSTM) preventing them from pursuing optimal conditions.

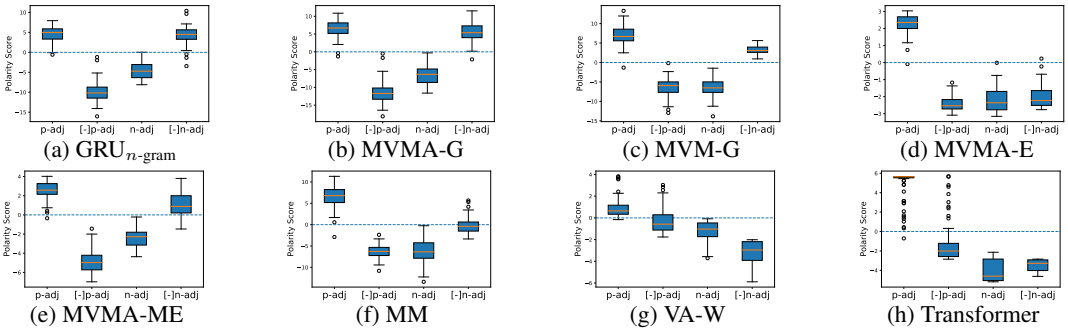
Figure 2e shows that the MVMA-ME model, which has a function A less complex than the ones from MVMA-G and MVMA-L but more complex than the one from MVMA-E, still can generally learn negation of negative adjectives better than the MVMA-E model. This demonstrates the necessity of choosing more expressive A and g functions.

Interestingly, both VA-W and Transformer are struggling with capturing the negation phenomenon for negative adjectives in our experiments as shown in Figures 2g and 2h, which suggests that they may have a weaker capability in modeling sequential features in our setup. However, we found they could still achieve good performances on the AG-news and IMDB datasets[13]. We hypothesize this is because the nature of SST-2 makes these two models suffer more on this dataset – it has rich linguistic phenomena such as negation cases while the other two datasets do not.

Additionally, we examined the ability for GRU, LSTM, MVMA-G and MVMA-L to capture both the negation and intensification phenomena. For such experiments, instead of using SST-2, we trained the models on SST-5, which comes with polarity intensity information. Polarity intensities were mapped into values of $\{-2, -1, 0, +1, +2\}$, ranging from *extremely negative* to *extremely positive*. We conducted some experiments based on the same setup above for capturing negation on SST-2. To our surprise, our preliminary results show that all models were performing substantially worse in terms of capturing intensification than capturing negations. We hypothesize that this is caused by the imbalance between negation phrases and intensification phrases. Specifically, the intensification word *"very"* (1,729 times) was exposed less than the negation word *"not"* (4,601 times) in the training set of SST-5.

One approach proposed in the literature for sentence classification is to consider all the hidden states of an RNN in an instance (Bahdanau et al., 2015). We believe this may actually be able to al-

[13]We conducted additional experiments for Transformers on sentiment analysis. Results are in appendix.



(a) GRU$_{n\text{-gram}}$ (b) MVMA-G (c) MVM-G (d) MVMA-E

(e) MVMA-ME (f) MM (g) VA-W (h) Transformer

Figure 2: Distribution of polarity scores for adjectives and their negation bigrams on SST-2. *p-adj* and *n-adj* refer to the positive and negative adjectives respectively. [-] refers to the negation operation (prepending the word "not"). Circles refer to outliers. More results can be found in the appendix.

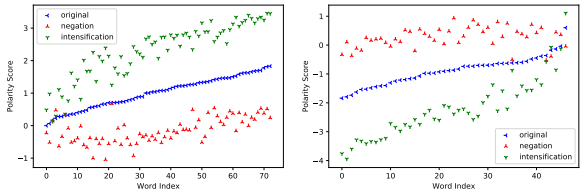


Figure 3: Context polarity scores (MVMA-G, SST-5) for positive (L) and negative (R) adjectives along with their negation and intensification bigrams.

leviate the above issue as it allows more n-grams within an instance to be exposed to the label information. Thus, we followed their approach for training our MVMA and MVM models[14].

We can see that the negation and intensification phenomena can be explained by both the context representations in Figure 3[15]. Specifically, prepending either positive or negative adjectives with "*very*" will likely strengthen their polarity while adding "*not*" will likely weaken their polarity. These results suggest that RNNs are able to capture information of linguistic significance within the sequence, and our identified n-gram representations within their hidden states appear to be playing a salient role.

5.3 Discussion

From the experiments above, we can see that our introduced n-gram representations, coupled with the corresponding context representations, are powerful in practice in capturing n-gram information better than the baseline compositional models introduced in the literature. We also found that RNNs can induce such representations due to their recurrence mechanism[16].

However, there can be several factors that affect the efficacy of different representations. First, through comparisons with different variants of MVMA, we can see that the precise way of parameterizing the functions $A(x_t)$ and $g(x_t)$ matter. Second, through the comparison between MVMA and MVM, we can see that defining an appropriate context representation that incorporates a correct set of n-grams is also important. Third, for models which do not capture such explicit n-gram features like ours, interestingly, they may still be able to yield good performances on certain tasks. For example, though VA-W and Transformer did not perform well on SST-2, they yielded results competitive to GRU and LSTM on AG-news and IMDB. This observation indicates there could be other useful features captured by such models that can contribute towards their overall modeling power.

Although in this work we did not aim to propose novel or more powerful architectures, we believe our work can be a step towards better understanding of RNN models. We also hope it can provide inspiration for our community to design more interpretable yet efficient architectures.

6 Conclusion

In this work, we focused on investigating the underlying mechanism of RNNs in terms of handling sequential information from a theoretical perspective. Our analysis reveals that RNNs contain a mechanism where each hidden state encodes a weighted combination of salient components, each of which can be interpreted as a representation of a classical n-gram. Through a series of comprehensive empirical studies on different tasks, we confirm our understandings on such interpretations of these components. With the analysis coupled with experiments, we provide findings on how RNNs learn to handle certain linguistic phenomena such as negation and intensification. Further investigations on understanding how the identified mechanism may capture a wider range of linguistic phenomena such as multiword expressions (Schneider et al., 2014) could an interesting future direction.

[14]However, for simplicity, in this work we only used the mean context representations (or hidden states) instead of a weighted sum of them.

[15]More results are in the appendix.

[16]We also visualized the context representations and n-gram representations in the appendix, which provide intuitive understanding of them.

Acknowledgements

We would like to thank the anonymous reviewers and our ARR action editor for their constructive comments. This research/project is supported by the Ministry of Education, Singapore, under its Tier 3 Programme (The Award No.: MOET32020-0004). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the Ministry of Education, Singapore.

References

Leila Arras, José Arjona-Medina, Michael Widrich, Grégoire Montavon, Michael Gillhofer, Klaus-Robert Müller, Sepp Hochreiter, and Wojciech Samek. 2019. Explaining and interpreting lstms. In *Explainable ai: Interpreting, explaining and visualizing deep learning*, pages 211–238. Springer.

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of ACL*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of EMNLP*.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. In *Proceedings of AAAI*.

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. 2020. Generating hierarchical explanations on text classification via feature interaction detection. In *Proceedings of ACL*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *Proceedings of ICLR*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159.

J. Elman. 1990. Finding structure in time. *Cogn. Sci.*, 14:179–211.

Melikasadat Emami, Mojtaba Sahraee-Ardakan, Parthe Pandit, Sundeep Rangan, and Alyson K Fletcher. 2021. Implicit bias of linear rnns. In *Proceedings of ICML*.

Rémi Eyraud and Stéphane Ayache. 2020. Distillation of weighted automata from recurrent neural networks using a spectral approach. https://arxiv.org/abs/2009.13101.

Gottlob Frege. 1948. Sense and reference. *The philosophical review*, 57(3):209–230.

Surya Ganguli, Dongsung Huh, and Haim Sompolinsky. 2008. Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences*, 105:18970 – 18975.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2018. Improving neural language models with a continuous cache. In *Proceedings of ICLR*.

Pankaj Gupta and Hinrich Schütze. 2018. LISA: Explaining recurrent neural network judgments via layer-wIse semantic accumulation and example to pattern transformation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. 2019. Modeling recurrence for transformer. In *Proceedings of NAACL*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent transformers. *https://arxiv.org/abs/2203.07852*.

Armand Joulin, Moustapha Cissé, David Grangier, Hervé Jégou, et al. 2017. Efficient softmax approximation for gpus. In *Proceedings of ICML*.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*.

Sekitoshi Kanai, Yasuhiro Fujiwara, and Sotetsu Iwamura. 2017. Preventing gradient explosions in gated recurrent units. In *Proceedings of NeurIPS*.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *http://arxiv.org/abs/1506.02078*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of NeurIPS*.

Gérard Lallement. 1979. *Semigroups and combinatorial applications*. John Wiley & Sons, Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL*.

Tao Lei. 2021. When attention meets fast recurrence: Training language models with reduced compute. In *Proceedings of EMNLP*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of NAACL*.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015a. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of ACL*.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015b. When are tree structures necessary for deep learning of representations? In *Proceedings of EMNLP*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of NAACL*.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of NAACL*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL*.

Niru Maheswaranathan and David Sussillo. 2020. How recurrent networks implement contextual processing in sentiment analysis. In *Proceedings of ICML*.

Niru Maheswaranathan, Alex H. Williams, Matthew D. Golub, S. Ganguli, and David Sussillo. 2019. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. In *Proceedings of NeurIPS*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *https://arxiv.org/abs/1609.07843*.

William Merrill. 2019. Sequential neural networks as automata. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*.

William Merrill, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A. Smith, and Eran Yahav. 2020. A formal hierarchy of RNN architectures. In *Proceedings of ACL*.

José Meseguer and Ugo Montanari. 1990. Petri nets are monoids. *Information and computation*, 88(2):105–155.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *https://arxiv.org/abs/1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NeurIPS*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. In *Proceedings of ICLR*.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of ICML*.

W. James Murdoch, Peter J. Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from LSTMs. In *Proceedings of ICLR*.

W. James Murdoch and Arthur Szlam. 2017. Automatic rule extraction from long short term memory networks. In *Proceedings of ICLR*.

Emin Orhan and Xaq Pitkow. 2020. Improved memory in recurrent neural networks with sequential non-normal dynamics. In *Proceedings of ICLR*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Guillaume Rabusseau, Tianyu Li, and Doina Precup. 2019. Connecting weighted automata and recurrent neural networks through spectral learning. In *Proceedings of AISTATS*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of ACL*.

Grzegorz Rozenberg and Arto Salomaa. 2012. *Handbook of Formal Languages: Volume 3 Beyond Words*. Springer Science & Business Media.

Sebastian Rudolph and Eugenie Giesbrecht. 2010. Compositional matrix-space models of language. In *Proceedings of ACL*.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: Running the MWE gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Chandan Singh, W James Murdoch, and Bin Yu. 2019. Hierarchical interpretations for neural network predictions. In *Proceedings of ICLR*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of ENMLP*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Xiaobing Sun and Wei Lu. 2020. Understanding attention for text classification. In *Proceedings of ACL*.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. 2019. LSTM networks can perform dynamic counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *Proceedings of EMNLP*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of ACL*.

Ronald J. Williams and Jing Peng. 1990. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. 2021. Self-attention networks can process bounded hierarchical languages. In *Proceedings of ACL-IJCNLP*.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of EMNLP*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of NeurIPS*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of ACL*.

A Dataset Statistics

The statistics of the sentiment analysis, relation classification and NER datasets are shown in Table 6. The language modeling datasets are obtained from Einstein.ai and the statistics are shown in Table 7.

Data	Train	Dev	Test	V.size	Max.len	Class
SST-2	98,794	872	1,821	17,404	54	2
IMDB	17,212	4,304	4,363	63,311	437	2
AG-news	110,000	10,000	7,600	85,568	212	4
SST-5	318,582	41,447	82,600	18,025	54	5
SemEval	7,000	1,000	2,717	27,115	91	10
CoNLL-2003	14,987	3,466	3,684	26,873	113	20

Table 6: Statistics of the sentiment analysis, relation classification and NER datasets. "V.size" indicates the vocabulary size and "Max.len" indicates the maximum length of the instances. "SemEval" refers to the SemEval 2010 Task 8 dataset for relation classification. For CoNLL-2003, "class" refers to the tag size.

We created the binary dataset SST-2 by extracting instances (including phrases) with polarity from the constituency parse trees in the original SST dataset (Socher et al., 2013). We merged the labels *extremely positive* and *positive* as *positive* and the labels *extremely negative* and *negative* as *negative*. We also extracted all the phrases in the constituency parse trees from the original dataset and created the 5-class dataset SST-5. The labels *extremely positive*, *positive*, *neutral*, *negative* and *extremely negative* were mapped into +2, +1, 0, -1, and -2 respectively.

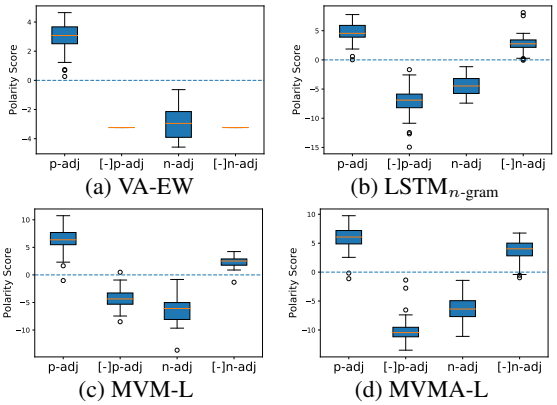
B More Result from the SST datasets

B.1 Negation and Intensification

Figure 4 shows that the n-gram representations from the LSTM model together with its corresponding MVMA-L and MVM-L models can also capture negation on the extracted adjectives from SST-2. However, VA-EW fails to capture the negation phenomenon for the negative adjectives, which may be explained by that: the n-gram representation of VA-EW solely involves the current token, thus being less expressive compared to the one from models such as MVMA-L and MVMA-G. Moreover, the MVMA-G model can also capture the negation and intensification phenomena on SST-5 as shown in Figure 5. The intensification token will generally strengthen the polarity of an adjective while the negation token will generally weaken the polarity of it.

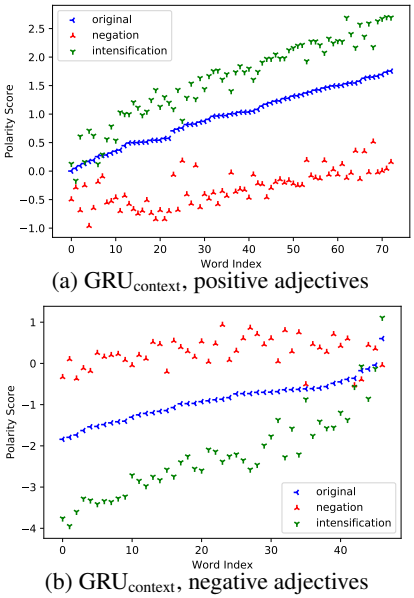
Dataset		Train	Dev	Test
PTB	Token Num	887,521	70,390	78,669
	Vocab Size		10,000	
Wiki2	Token Num	2,088,628	217,646	245,569
	Vocab Size		33,278	
Wiki103	Token Num	103,227,021	217,646	245,569
	Vocab Size		267,735	

Table 7: Statistics of the language modeling datasets.



(a) VA-EW (b) LSTM$_{n\text{-gram}}$ (c) MVM-L (d) MVMA-L

Figure 4: Distribution of polarity scores for adjectives and their negation bigrams. *p-adj* and *n-adj* refer to the positive and negative adjectives respectively. [-] refers to the negation operation (prepending the word "not"). Circles refer to outliers.

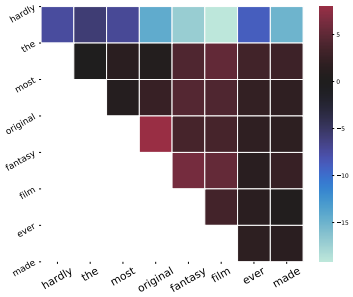


(a) GRU$_{\text{context}}$, positive adjectives

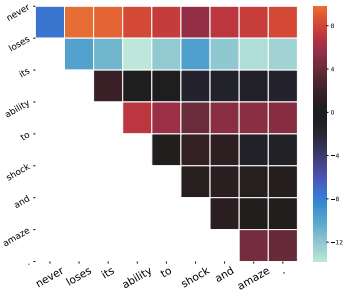
(b) GRU$_{\text{context}}$, negative adjectives

Figure 5: Context polarity scores for positive adjectives (a) and negative adjectives (b) along with their corresponding negation and intensification bigrams from SST-5.

We also visualized the polarity score of each n-gram within a sentence. Two examples are shown in Figures 6a and 6b, where a warmer color indicates a higher polarity score (i.e., the n-gram is more positive). For example, "*never*" itself has a remarkably negative polarity score while "*loses*" has a remarkably positive one. Consequently, the n-grams starting from "*never*" (while ending with another word) generally have positive polarity scores.



(a)



(b)

Figure 6: Polarity scores for n-gram representations within two example sentences. SST-2, MVMA-G.

Such visualization results show that our identified representations defined over the linguistic units as captured by RNNs can be highly interpretable.

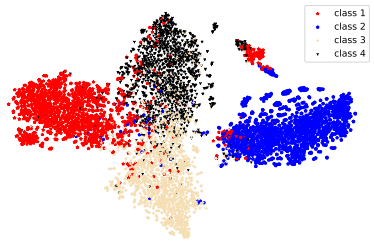
B.2 First-order Approximation

To examine how well the recurrence relation in Equation 7 can approximate the standard RNNs, we followed the method in the work of Maheswaranathan and Sussillo (2020) and compared the hidden state of the standard RNNs ($\boldsymbol{h}_t = RNN(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$) at each time step to the corresponding context representations ($\hat{\boldsymbol{h}}_t = \boldsymbol{g}(\boldsymbol{x}_t) + \boldsymbol{A}(\boldsymbol{x}_t)\boldsymbol{h}_{t-1}$). The error at each time step is defined as

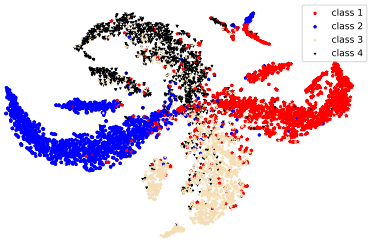
$$||\boldsymbol{h}_t - \hat{\boldsymbol{h}}_t||_2 / ||\boldsymbol{h}_t||_2. \quad (15)$$

We used the current standard hidden state to predict the next hidden state and the context representations on the SST-2 test set.

We noticed that the weight decaying coefficient has a remarkable impact on the error. Specifically, a larger coefficient can result in smaller errors. When the coefficient is $1e{-}5$, the average errors on the Elman, GRU, and LSTM models were 26.2%, 21.7% and 46.6% and respectively. When the coefficient is $3e-4$ the the average errors dropped to 17.1%, 15.1%, and 33.3% respectively. Note that since this is the single step error, the accumulated errors across many times steps can be large, particularly for LSTM, and thus the first-order approximation cannot fully replace standard RNNs. Despite this, the resulting context and n-gram representations can help us understand how RNNs process contextual information such as n-gram features.



(a) MVMA-G



(b) MVM-G

Figure 7: (a) and (b): T-sne visualization of the context representation for phrases (<30 tokens) from the AG-news dataset with four topics.

C T-sne Visualization

We visualized the context representations from the MVMA-G model using t-sne (van der Maaten and Hinton, 2008), which provides us with an intuitive understanding on the efficacy of our identified representations. We automatically extracted 2,188 phrases with less than 30 tokens from AG-news with 4 topics[17] and projected their context representations to a two-dimension space. Figures 7a and 7b show there exist four major clusters corresponding to the four topics, indicating those representations can generally learn the topic information and explain the differences. Similar to the previous analysis, the MVM-G model is able to learn the topic information with the n-gram representations.

D Results on Transformer

We have also run the Transformer model on the sentiment analysis datasets and the results are listed in Table 8.

[17] Although SST-5 has 5 lables, most of its phrases are neutral, we therefore did not use this dataset for visualization.

SST-2		AG-news		IMDB	
dev	test	dev	test	dev	test
83.4±0.4	82.0±0.1	90.9±0.5	90.5±0.4	88.4±0.2	88.1±0.2

Table 8: Accuracy on sentiment analysis tasks. Transformer

E Implementation Details

E.1 Sentiment Analysis

Settings For the SST-2, AG-news, and IMDB datasets, we used the cross-entropy as the loss function to train the models. Embeddings were randomly initialized and trainable during training. For the SST-5 dataset, we treated the classification as a regression problem as the labels are polarity intensity. The mean-squared error was used as the loss function during training. Note that we initialized embeddings with pre-trained GloVe (Pennington et al., 2014) and fixed them during training on SST-5 for the analysis of both the negation and intensification phenomena.

Furthermore, for the MM model, each token was represented as a matrix and the matrix size was set as 32×32. For the other models, the embedding and hidden sizes were both set as 300.

Polarity Adjectives We automatically extracted adjectives with polarity (examples shown in Table 9) from SST-2 in two steps. In the first step, following the method of Sun and Lu (2020), we calculated a frequency ratio for each token (in the vocabulary) between the frequencies of the token seen in the positive and negative instances respectively. If a token has a frequency ratio either larger than 3 or less than 1/3, it will be extracted as an *positive token* or an *negative token*. In the second step, we used the *textblob* package [18] to detect positive and negative adjectives from those positive tokens and negative tokens respectively.

E.2 Relation Classification

Following the work of Gupta and Schütze (2018), we examined the RNN, baseline, MVMA and MVM models on SemEval 2010 Task 8 (Hendrickx et al., 2010) which has 9 directed relationships and an undirected *other* type. We used the final hidden states of the standard RNNs (or context representations of the MVMA, MVM and baseline models) as the instance representations for classification. The cross-entropy loss was employed during training.

[18] https://textblob.readthedocs.io/en/dev/

E.3 Named Entity Recognition

At each time step, we concatenated the context representations (or hidden states) from both directions in a bidirectional model, fed them to a projection layer and then to a linear CRF layer. More details about the architecture can be referred to the biLSTM-CRF model in the work of Lample et al. (2016). We also referred to the code at https://github.com/allanj/pytorch_neural_crf for the implementation of the linear CRF layer.

CoNLL-2003 contains four types of entities: persons (PER), organizations (ORG), locations (LOC) and miscellaneous names (MISC). The original dataset was labeled with the BIO (Beginning-Inside-Outside) format. For example, "United Arab Emirates" are labeled as "B-LOC I-LOC I-LOC". We transform the tags into the IOBES format where two prefixes "E-" and "S-" are added. Specifically, "E-" is used to label the last token of an entity span. The "S-" prefix is used for a single-token span. For example, "United Arab Emirates" are labeled as "B-LOC I-LOC E-LOC" in this format. There are 20 categories of tags in total including the starting, ending and padding tags. We trained the models to predict each entity.

The embedding size and hidden size were set to 300 and 200 respectively. The SGD optimizer was used to learn parameters.

E.4 Language Modeling

The embedding size and hidden size were both 512 for PTB and Wiki2, and 256 and 512 respectively for Wiki103. The cross-entropy loss was used during training. For PTB and Wiki2, the output of the final fully-connected layer was fed to a *softmax* function while the Adaptive softmax (Joulin et al., 2017) was used for Wiki103 (because of its large vocabulary size). We only considered the word-level models. We trained each model for 50 epochs, chose the model which had the best performance on the development set as the final model and evaluated the final model on the test set.

F Jacobian matrix of LSTM

Unlike GRU and Elman RNN, LSTM has a memory cell apart from a hidden state. Here, we describe how to get their Jacobian matrices. An

Type	Adjectives	Size
Pos	outstanding, ecological, inventive, comfortable, nice, authentic, spontaneous, sympathetic, lovable, unadulterated, controversial, suitable, grand, happy, enthusiastic, adventurous, successful, noble, true, detailed, sophisticated, sensational, exotic, fantastic, remarkable, impressive, charismatic, good, effective, rich, popular, unforgettable, famous, comical, energetic, ingenious, extraordinary, ...	73
Neg	bad, tedious, miserable, psychotic, didactic, inexplicable, feeble, sloppy, disastrous, stupid, amateurish, false, cynical, farcical, terrible, unhappy, horrible, atrocious, idiotic, wrong, pathetic, angry, uninspired, vicious, unfocused, unnecessary, artificial, troubled, questionable, arduous, stereotypical, ...	47

Table 9: Examples of the extracted adjectives from the SST-2 dataset. "Pos" refers to *positive* adjectives and "Neg" refers to *negative* adjectives.

LSTM cell can be written as

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{ii}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1}),\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{if}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1}),\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{io}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1}),\\
\boldsymbol{c}_t^m &= \tanh(\boldsymbol{W}_{ic}\boldsymbol{x}_t + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1}),\\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{c}_t^m, \boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t),
\end{aligned} \tag{16}$$

where $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{o}_t \in \mathbb{R}^d$ are the input gate, forget gate and output gate respectively. $\boldsymbol{c}_t^m \in \mathbb{R}^d$ is the new memory, and $\boldsymbol{c}_t$ is the final memory.

Let us expand the memory state and hidden state at time step t as

$$\begin{aligned}
\boldsymbol{c}_t &= \boldsymbol{g}_c(\boldsymbol{x}_t) + B(\boldsymbol{x}_t)\boldsymbol{c}_{t-1}\\
&\quad + D(\boldsymbol{x}_t)\boldsymbol{h}_{t-1} + \boldsymbol{o}_c(\boldsymbol{c}_{t-1}, \boldsymbol{h}_{t-1}),\\
\boldsymbol{h}_t &= \boldsymbol{g}_h(\boldsymbol{x}_t) + E(\boldsymbol{x}_t)\boldsymbol{c}_{t-1}\\
&\quad + F(\boldsymbol{x}_t)\boldsymbol{h}_{t-1} + \boldsymbol{o}_h(\boldsymbol{c}_{t-1}, \boldsymbol{h}_{t-1}),
\end{aligned} \tag{17}$$

where B, D, E and $F \in \mathbb{R}^{d\times d}$ are all Jacobian matrices. $\boldsymbol{o}_c(\boldsymbol{h}_{t-1})$ and $\boldsymbol{o}_h(\boldsymbol{h}_{t-1})$ are remainder terms of the Taylor expansion.

We concatenate the memory state and hidden state and view the concatenation as an "extended hidden state". The context representation for the "extended hidden state" at time step t (assuming of zero vectors as initial states) will be written as:

$$\begin{bmatrix}\hat{\boldsymbol{c}}_t\\ \hat{\boldsymbol{h}}_t\end{bmatrix} = \sum_{i=1}^{t}\begin{bmatrix}\boldsymbol{v}_{i:t}^c\\ \boldsymbol{v}_{i:t}^h\end{bmatrix} = \sum_{i=1}^{t}\left[\prod_{k=t}^{i+1} A(x_k)\right]\begin{bmatrix}g_c(\boldsymbol{x}_i)\\ g_h(\boldsymbol{x}_i)\end{bmatrix}, \tag{18}$$

where $\hat{\boldsymbol{c}}_t$ and $\hat{\boldsymbol{h}}_t$ refer to the context representations corresponding to the memory state and hidden state respectively. g_c, $g_h \in \mathbb{R}^d$, and $A \in \mathbb{R}^{2d\times 2d}$ are all functions of inputs. $A(x_k)$ contains many interaction terms resulting from the gating mechanism, which may result in a strong expressive power. As the hidden state $\boldsymbol{h}_t$ is commonly used for downstream tasks, we will only consider $\boldsymbol{v}_{i:t}^h$ as the n-gram representation on our tasks, and the context representation will be $\sum_{i=1}^{t}\boldsymbol{v}_{i:t}^h$.