How recurrent networks implement contextual processing in sentiment analysis

Niru Maheswaranathan [* 1]   **David Sussillo** [* 1]

Abstract

Neural networks have a remarkable capacity for contextual processing—using recent or nearby inputs to modify processing of current input. For example, in natural language, contextual processing is necessary to correctly interpret negation (e.g. phrases such as "not bad"). However, our ability to understand *how* networks process context is limited. Here, we propose general methods for reverse engineering recurrent neural networks (RNNs) to identify and elucidate contextual processing. We apply these methods to understand RNNs trained on sentiment classification. This analysis reveals inputs that induce contextual effects, quantifies the strength and timescale of these effects, and identifies sets of these inputs with similar properties. Additionally, we analyze contextual effects related to differential processing of the beginning and end of documents. Using the insights learned from the RNNs we improve baseline Bag-of-Words models with simple extensions that incorporate contextual modification, recovering greater than 90% of the RNN's performance increase over the baseline. This work yields a new understanding of how RNNs process contextual information, and provides tools that should provide similar insight more broadly.

1. Introduction

Neural networks do a remarkable job at learning structure in natural data. These architectures exploit complex relationships between inputs to perform tasks at state-of-the-art levels (LeCun et al., 2015). Despite this amazing performance, we still only have a rudimentary understanding of exactly how these networks work (Castelvecchi, 2016).

*Equal contribution [1]Google Research, Brain Team, Mountain View, California, USA. Correspondence to: Niru Maheswaranathan <nirum@google.com>, David Sussillo <sussillo@google.com>.

Rigorously understanding how networks solve important tasks is a central challenge in deep learning. This understanding is lacking because we use complex optimization procedures to set the parameters of ever more complex network architectures. Our inability to fundamentally understand how a trained system works makes it difficult to identify biases in the network or training data, control for adversarial input, bracket or bound network behavior, suggest ways of improving efficiency or accuracy, and elucidate the (potentially simple) core mechanisms underlying the task.

In this work, we focus on building general tools and analyses to understand how recurrent neural networks (RNNs) process contextual effects. By *contextual effects*, we mean effects where the interpretation of an input depends on surrounding inputs. For example, in natural language, the interpretation of words is modified by preceding words for negation ("not bad" vs "bad") or emphasis ("extremely awesome" vs "awesome") (Quirk et al., 1985; Horn & Kato, 2000). We analyze networks trained to perform sentiment classification, a commonly studied natural language processing (NLP) task (Wiegand et al., 2010; Mohammad, 2017; Zhang et al., 2018).

When interpreting or understanding how recurrent networks work, one line of research uses attribution methods to analyze sensitivity to particular inputs (Simonyan et al., 2013; Ribeiro et al., 2016; Li et al., 2015; Arras et al., 2017; Murdoch et al., 2018). These methods utilize gradients of the network loss or unit activations with respect to inputs. A second line of research uses tools from dynamical systems analysis to decompose RNN trajectories (Zipser, 1991; Tsung & Cottrell, 1995; Casey, 1996; Rodriguez et al., 1999; Sussillo & Barak, 2013; Mante et al., 2013; Cueva & Wei, 2018; Jordan et al., 2019; Maheswaranathan et al., 2019b). Here, we merge these lines of research to understand contextual processing of large numbers of inputs in recurrent systems.

In particular, our key insight is that rich contextual processing can be understood as a composition of precisely organized input-driven state deflections combined with highly organized transient dynamics. Critically, the former can be as understood using Jacobians with respect to the input, analogous to attribution methods, while the latter can be understood through Jacobians with respect to the recurrent state, analogous to work on RNN dynamics.
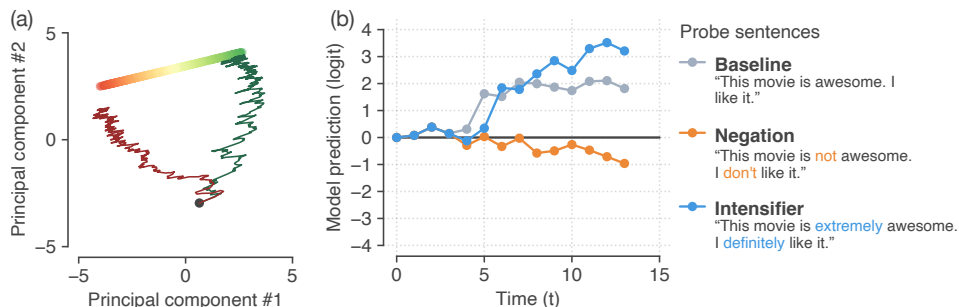
Figure 1. RNNs and contextual processing. (a) Approximate line attractor dynamics in a GRU trained to perform sentiment classification. PCA visualization of trajectories of the system during an example positive (green) and negative (red) review. The approximate line attractor is visualized as the light red to green line, colored according to the readout (logit). (b) Evolution of RNN predictions (logits) when processing example reviews over time (tokens in the input sentence). Gray (baseline): "This movie is awesome. I like it" (with pad tokens inserted to align it with the other examples), Blue (intensifier): "This movie is extremely awesome. I definitely like it." Orange (negation): "This movie is not awesome. I don't like it." The RNN correctly handles the intensifier and negation context.

Our main contributions in this work are:

- a data-driven method for identifying contextual inputs,
- a breakdown of the types of contextual effects learned,
- analysis of the strength and timescale of these effects,
- a mathematical model that describes this behavior,
- and a demonstration that simple and interpretable baseline models, augmented to incorporate this new understanding, recover almost all of the accuracy of the nonlinear RNNs.

2. Background

Previous work (Maheswaranathan et al., 2019b) analyzed recurrent networks trained to perform sentiment classification. They found that RNNs learned to store the current prediction as the location along a 1D manifold of approximate fixed points of the dynamics, called a *line attractor* (Seung, 1996; Mante et al., 2013). Additionally, they found that words with positive and negative valence drive the hidden state along this line attractor, which is aligned with the readout. A recap of this analysis is presented in Figure 1a, for a gated recurrent unit (GRU) (Cho et al., 2014) trained using the Yelp 2015 dataset (Zhang et al., 2015).

Critically, the mechanisms discussed in Maheswaranathan et al. (2019b) involved only integration of valence tokens along the line attractor. We tested the RNN on the same task after randomly shuffling the input words. Shuffling breaks apart important contextual phrases, such as "really awesome" and "not bad". On the shuffled test examples, the accuracy drops by $\sim 4\%$ (Fig. 10a), close to the performance of a Bag-of-Words (BoW) baseline model. Thus, there are additional accuracy gains, present in the best performing RNNs, that are not explainable using prior known mechanisms.

The capability of RNNs to understand context can also be observed by probing the RNN with examples that contain contextual effects. Figure 1b shows the predictions of the RNN in response to three probe reviews, a baseline ("This movie is awesome. I like it."), one with **intensifiers** ("This movie is **extremely** awesome. I **definitely** like it."), and one with **negation** ("This movie is **not** awesome. I **don't** like it."). The RNN is capable of correctly assessing the sentiment in these reviews. Below, we break down exactly how the RNN is able to accomplish this.

3. Preliminaries

3.1. Linearization and expansion points

We denote the hidden state of a recurrent network at time t as a vector, $\mathbf{h}_t$. Similarly, the input to the network at time t is given by a vector $\mathbf{x}_t$. We use F to denote a function that applies the recurrent network update, i.e. $\mathbf{h}_t = F(\mathbf{h}_{t-1}, \mathbf{x}_t)$. The RNN defines an input-driven discrete-time dynamical system that sequentially processes inputs, in this case words in a document encoded as a sequence of one-hot input vectors. The final prediction (logit) is an affine projection (or readout) of the hidden state. In this work, we focus on binary sentiment classification, thus the logit is a scalar ($\mathbf{w}^T \mathbf{h}_t + b$), with readout weights $\mathbf{w}$ and bias b.

We can write the first-order approximation to the RNN dynamics (Khalil, 2001; Maheswaranathan et al., 2019b) around an expansion point $(\mathbf{h}^{\mathrm{e}}, \mathbf{x}^{\mathrm{e}})$ as:

$$\mathbf{h}_t \approx F(\mathbf{h}^{\mathrm{e}}, \mathbf{x}^{\mathrm{e}}) + \mathbf{J}^{\mathrm{rec}}|_{(\mathbf{h}^{\mathrm{e}}, \mathbf{x}^{\mathrm{e}})} \Delta \mathbf{h}_{t-1} + \mathbf{J}^{\mathrm{inp}}|_{(\mathbf{h}^{\mathrm{e}}, \mathbf{x}^{\mathrm{e}})} \Delta \mathbf{x}_t, \quad (1)$$

where $\Delta \mathbf{h}_{t-1} = \mathbf{h}_{t-1} - \mathbf{h}^{\mathrm{e}}$, $\Delta \mathbf{x}_t = \mathbf{x}_t - \mathbf{x}^{\mathrm{e}}$, and $\{\mathbf{J}^{\mathrm{rec}}, \mathbf{J}^{\mathrm{inp}}\}$ are Jacobian matrices computed at the expansion point. In particular, the **recurrent Jacobian** $\left(J^{\mathrm{rec}}_{ij}|_{(\mathbf{h}^{\mathrm{e}}, \mathbf{x}^{\mathrm{e}})} = \frac{\partial F(\mathbf{h}, \mathbf{x})_i}{\partial h_j} \right)$ defines the recurrent local dynamics and the **input Jacobian** $\left(J^{\mathrm{inp}}_{ij}|_{(\mathbf{h}^{\mathrm{e}}, \mathbf{x}^{\mathrm{e}})} = \frac{\partial F(\mathbf{h}, \mathbf{x})_i}{\partial x_j} \right)$ defines the system's sensitivity to inputs.

3.2. Linearization to understand dynamics

Fixed points are points in state space that remain the same when applying the RNN: $\mathbf{h}^* = F(\mathbf{h}, \mathbf{x}^*)$. If the expansion point $(\mathbf{h}^e, \mathbf{x}^e)$ is an approximate fixed point of the dynamics, equation (1) simplifies to a *linear* dynamical system:

$$\Delta\mathbf{h}_t \approx \mathbf{J}^{\mathrm{rec}}|_{(\mathbf{h}^*, \mathbf{x}^*)}\Delta\mathbf{h}_{t-1} + \mathbf{J}^{\mathrm{inp}}|_{(\mathbf{h}^*, \mathbf{x}^*)}\Delta\mathbf{x}_t. \quad (2)$$

We use equation (2) to study integration dynamics around approximate fixed points, as in Maheswaranathan et al. (2019b). These approximate fixed points are found numerically (see Supp. Mat. §B for details).

3.3. Linearization to understand modifier words

To study the effects of modifier tokens, we also need to analyze the system *away* from approximate fixed points. In particular, we analyze the system at $\mathbf{h}^{\mathrm{mod}}$, defined as the state after processing a particular modifier word. To do this, we linearize with respect to just the inputs for a single time step, expanding only in $\mathbf{x}$, around $\mathbf{x}^e = \mathbf{0}$:

$$\Delta\mathbf{h}_t = \mathbf{h}_t - F(\mathbf{h}^{\mathrm{mod}}, \mathbf{0}) \approx \mathbf{J}^{\mathrm{inp}}|_{(\mathbf{h}^{\mathrm{mod}}, \mathbf{0})}\mathbf{x}_t, \quad (3)$$

where equation (3) does not expand in $\mathbf{h}$, to focus on input sensitivity. We make extensive use of the input Jacobian in equation (3) in our analysis in §5.

4. Toy language to isolate modifier dynamics

In order to illuminate how negation or emphasis is implemented in RNNs we developed a small toy language to isolate modifier effects. The language consisted of a small number of valence tokens, each with integer valence $\{-2, -1, 0, 1, 2\}$, analogous to words such as "awful", "bad", "the", "good", and "awesome", respectively. In addition, we added two modifier words, an intensifier that doubled the valence of the next input, and a negator that flipped the sign of the valence of the next four inputs (analogous to words such as "**extremely**" and "**not**").

We generated reviews using this language by randomly ordering the tokens and trained RNNs to track the corresponding sentiment, defined as the cumulative sum of potentially modified valence inputs. For example, after training, the RNN correctly integrated "good" as +1, "**extremely** good" as +2 and "**not** good" as -1. We analyzed the networks using the methods developed by Maheswaranathan et al. (2019b). An example state-space plot from a trained network is shown in Figure 2. Note that this RNN also exhibits line attractor dynamics.

We draw two key insights from this exercise (see Supp. Mat. §A for a full analysis). First, modifiers achieve their effects by deflecting the state *away* from the line attractor as opposed to along it, the latter of which is what valence
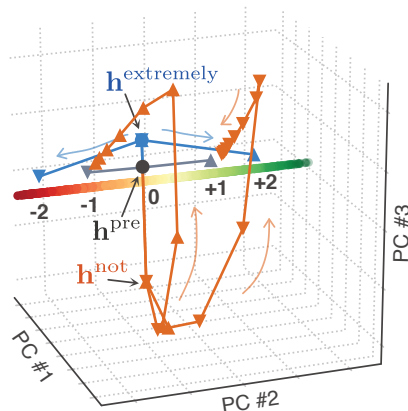

Figure 2. State-space plot of a GRU trained on a toy language to isolate effects of modifier tokens. This example shows a line attractor (line from red to green) that performs integration of valence (value from -2 to +2, bold black numbers). Modifiers such as "**extremely**" (blue) and "**not**" (orange) achieve their effects by deflecting the state away from the line attractor to $\mathbf{h}^{\mathrm{extremely}}$ and $\mathbf{h}^{\mathrm{not}}$, respectively. The effect of these deflections is to modify the valence of the input when projected onto the readout. Six trajectories are shown, all starting from $\mathbf{h}^{\mathrm{pre}}$. Gray shows the single step trajectory for "good" (▲, +1) and "bad" (▼, -1). In blue are the single-step trajectories for "**extremely** good" (▲, +2) and "**extremely** bad" (▼, -2), and the multi-step trajectories for "**not** good" (▲, -1) and "**not** bad" (▼, +1) are in orange. We defined the duration of the effect of the "**not**" modifier to be 4 tokens.

words do. We found that the effect of this deflection away from the line attractor on the valence of the subsequent word was very well approximated by $\mathbf{w}^T \mathbf{J}^{\mathrm{inp}}|_{(\mathbf{h}^{\mathrm{mod}}, \mathbf{0})} \mathbf{x}^{\mathrm{val}}$ (Supp. Fig. 12), where $\mathbf{h}^{\mathrm{mod}}$ is the state after the modifier input, $\mathbf{x}^{\mathrm{val}}$ is the valence word following the modifier, and $\mathbf{w}$ and bias b are the readout weights and bias. As shown in the Supp. Mat. §A, removing the projection of $\mathbf{J}^{\mathrm{inp}}|_{(\mathbf{h}^{\mathrm{mod}}, \mathbf{0})} \mathbf{x}$ into the modification subspace removes the effect of the modifier (Supp. Fig. 14).

Second, the deflection of the state away from the line attractor caused by $\mathbf{x}^{\mathrm{mod}}$ remains for the duration of the modifier effect. For example, in the toy language we confined the temporal extent of the modification effects of "extremely" and "not" to one word and four words, respectively, and the corresponding deflections off the line attractor remain for one and four time steps, respectively. Finally, the transient dynamics associated with valence modification can be isolated in the local linear dynamics around fixed points on the line attractor (Supp. Fig. 11 & 13).

5. Reverse engineering contextual processing

We turn our attention now to natural language, studying our best performing RNN, a GRU, trained to perform sentiment classification on the Yelp 2015 dataset (Zhang et al., 2015).
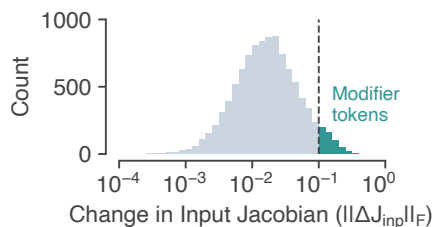
5.1. Identifying modifier words

Figure 3. Identifying modifier words. Histogram of Frobenius norm of change in input Jacobian. Note the log scaling on the x-axis, implying this distribution is heavy-tailed. We defined a modifier token as anything with a norm greater than 0.1. Example words above this threshold that are intuitively modifiers include "not", "never", "overall", and "definitely". Additional words found by this measure that are not as intuitive include: "zero", "two", "poisoning", and "worst".

For RNNs trained on real world datasets, we did not know what words might act as modifiers, therefore, we wanted a data driven approach to identify them. Inspired by the toy model, we looked for particular words that deflected the hidden state to dimensions where the input Jacobian changed substantially, as that indicated differential processing of inputs. We defined the change in input Jacobian after a particular input, $\mathbf{x}^{\mathrm{mod}}$, as:

$$\Delta \mathbf{J}^{\mathrm{inp}}|_{\mathbf{h}^{\mathrm{mod}}} \equiv \frac{dF}{d\mathbf{x}}\bigg|_{(\mathbf{h}^{\mathrm{mod}},\mathbf{0})} - \frac{dF}{d\mathbf{x}}\bigg|_{(\mathbf{h}^*,\mathbf{0})}, \qquad (4)$$

where $\mathbf{h}^{\mathrm{mod}} = F(\mathbf{h}^*, \mathbf{x}^{\mathrm{mod}})$ is the hidden state after processing $\mathbf{x}^{\mathrm{mod}}$, starting from a point on the line attractor, $\mathbf{h}^{*}$[1]. Thus, $\Delta \mathbf{J}^{\mathrm{inp}}|_{\mathbf{h}^{\mathrm{mod}}}$ measures how the system's processing of words changes as a function of a a preceding word, $\mathbf{x}^{\mathrm{mod}}$.

We studied the size of the changes in input processing by computing the Frobenius norm, $\|\Delta \mathbf{J}^{\mathrm{inp}}|_{\mathbf{h}^{\mathrm{mod}}}\|_F$, for each of the top 10,000 most common words in the dataset. The resulting distribution is shown in Figure 3 and is evidently heavy-tailed. Examining words with large values reveals common negators (e.g. "not", "never") and intensifiers (e.g. "very", "definitely"). We selected a set of modifier words for further analysis by keeping words whose change in input Jacobian was greater than an arbitrary threshold of 0.1[2].

5.2. Analyzing example modifier words

5.2.1. DYNAMICS OF EXAMPLE MODIFIER TRANSIENTS

We next looked at the dynamics of the hidden state in response to example modifier words, "**extremely**" or "**not**",

---

[1] We suppress the $\mathbf{h}^*$ in the notation as it was held constant in our analyses.

[2] None of the presented results are particularly sensitive to this choice.
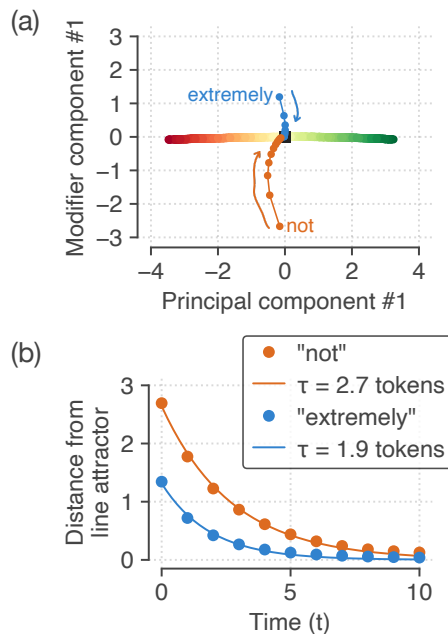

Figure 4. Impulse response to a modifier word. (a) Two trajectories of the network hidden state in response to modifier words "**extremely**" and "**not**", projected onto the top PCA component (x-axis) as well as the top modifier component (y-axis); see text for details. (b) Distance from the line attractor after a single modifier word. Circles show response of the RNN, line is an exponential fit.

followed by a series of pad tokens. Figure 4a shows the impulse response to "**extremely**" and "**not**", projected into a two-dimensional subspace for visualization[3]. Each modifier word deflects the hidden state off of the line attractor, which then relaxes back over the course of 5-10 time steps, similarly to the toy example. Figure 4b shows the same impulse response as a function of time, quantified using the Euclidean distance from the line attractor. These transients, induced by modifiers, explore parts of the RNN state space that critically are not contained in the top two PCA dimensions (Fig 1a). It was only through focusing on the change in input Jacobian that we were able to identify them.

5.2.2. MODIFIER BARCODES

Next, we asked *how* the system leverages these transient dynamics to enable contextual processing. To answer this, we analyzed how the input Jacobian changes along these transients in comparison to their values at the line attractor. However, a complication arises because a modifier may have differing effects on each word and there are many words in the vocabulary. Therefore we developed a visualization, a **modifier barcode**, to study the effect of changing Jacobians.

---

[3] The two-dimensional subspace in Fig 4 consists of the top PCA component (x-axis) and the top modifier component (defined later in §5.3.1).
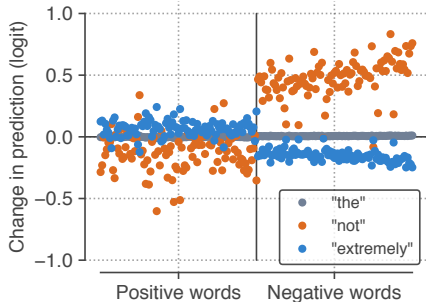
Figure 5. Barcodes for visualizing the effects of modifiers. The barcode is a quantitative signature of a particular modifier word. Shown are the barcodes for 'the' (gray), "**extremely**" (blue), and "**not**" (orange). Each point in a barcode is the change in the model's prediction in response to a particular valence word when a modifier word precedes it (e.g. "not great", "not bad", "not amazing", etc. (orange)), for the top 100 positive words (e.g. "amazing", "awesome"; left) and the top 100 most negative words (e.g. "hate", "awful"; right). For example the "not" barcode shows a negative or reduced change for positive valence words and a strong positive output change for negative words.

Figure 6. Low-dimensional modifier subspace. Projection of modifier points (teal circles) onto the top two modifier components, obtained by performing principal components analysis (PCA) on the set of states after processing modifier words. Highlighted are a number of example modifiers, including "**not**" and "**extremely**") from the previous figure. Emphasizers (e.g. "extremely", "very") are on one side of the subspace, whereas negators (e.g. "not", "never) are on the other side.

Barcodes are intended to quantify how a given modifier word affects processing of future inputs. For a given probe word $\mathbf{x}$, we define a barcode for a modifier word as:

$$\text{barcode}^{\text{mod}}(\mathbf{x}) = \mathbf{w}^T \Delta \mathbf{J}^{\text{inp}}|_{\mathbf{h}^{\text{mod}}} \mathbf{x}, \qquad (5)$$

where $\mathbf{h}^{\text{mod}}$ is the RNN state after a modifier word has been processed. As there were 90,000 words in the vocabulary we instead selected 100 positive and 100 negative words (e.g. "awesome" and "awful") to comprise the barcode[4].

Figure 5 shows three barcodes, for the words "the", "**extremely**", and "**not**". For words that are not modifiers (such as "the"), the effects of inputs before and after the word are the same, thus the barcode values are close to zero. For intensifiers, such as "**extremely**", we see that positive words (left half of Figure 5) and negative words (right half) are accentuated). For negators, such as "**not**", positive and negative words are made relatively more negative or more positive, respectively. In summary, the modifier barcode allows us to easily assess *what* the effects of a particular modifier are, through the changing input Jacobian.

5.3. Summary across all modifier words

5.3.1. MODIFIER SUBSPACE

So far we have analyzed the modifier dynamics and barcodes for a couple of example modifiers. Next, we will summarize

these effects over the entire set of identified modifier words. In particular, we are interested in understanding how many different types of modifiers are there, and whether we can succinctly describe their effects and dynamics.

First, we collected all of the deflections in the hidden state induced by modifier words. This is a set of points in the RNN hidden state where there are substantial changes to the input Jacobian. We ran principal components analysis on this set of points to identify a low-dimensional *modifier subspace*, also ensuring that the subspace is orthogonal to the line attractor[5]. In particular, two components explained 96.2% of the variance, suggesting that the dominant modifier effects can be understood as occurring in a 2D plane.

We projected all of the modifier inputs into this 2D subspace for visualization to highlight the distribution of modifiers in this subspace, as well as some example modifiers (Fig. 6). Negators (words like "not" and "never") live in one part of the subspace, whereas intensifiers (words like "extremely" and "definitely") live in another part. Interestingly, these two types of modifiers share the same subspace.

5.3.2. FAST AND SLOW MODIFIER DYNAMICS

We then studied the dynamics in the modifier subspace. In particular, the dynamics of how the hidden state evolves in this subspace determines the length of the effects of each modifier (called the "scope" in linguistics (Quirk et al., 1985; Horn & Kato, 2000)). Figure 7 shows the dynamics of mod-

---

[4]Words were selected by taking the 100 largest (most positive) and 100 smallest (most negative) logistic regression weights, using a separately fit logistic regression classifier (Bag-of-words model). Results do not depend on the particular set of probe words chosen.
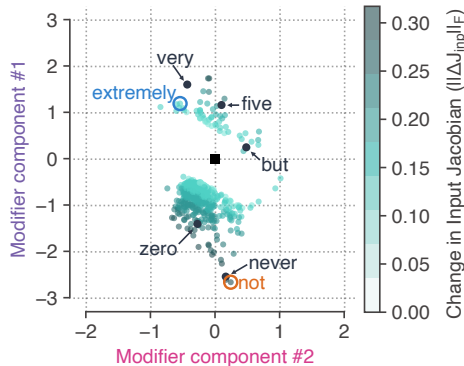
[5]We do this additional orthogonalization step to ensure that the identified modifier subspace does not have any valence effects, which occur along the line attractor.
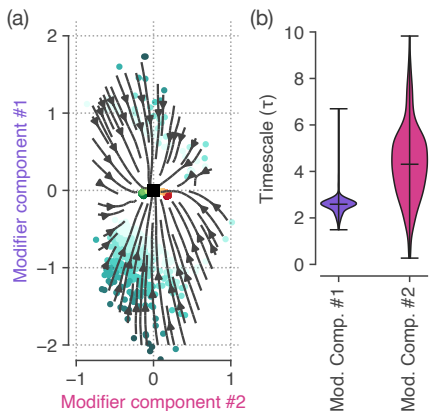
Figure 7. Timescales of modifier effects. (a) Hidden state dynamics of the impulse response to modifier inputs. Note the line attractor projects out of the page. (b) Distribution of the estimated timescale of the decay along each modifier component across modifier words.

ifiers in the 2D subspace, along with a quantification of the timescale of the decay in each modifier dimension or component (Fig. 7b). We observed that the first modifier component is faster (mean τ = 2.6 tokens), whereas the second one is slower (mean τ = 4.3 tokens), though there is spread in the distributions in both dimensions. This argues that the scope of modifier words in RNNs trained to perform sentiment classification lasts for tens of tokens and agrees with estimates of the scope of negation from human annotators (Taboada et al., 2011; Strohm & Klinger, 2018).
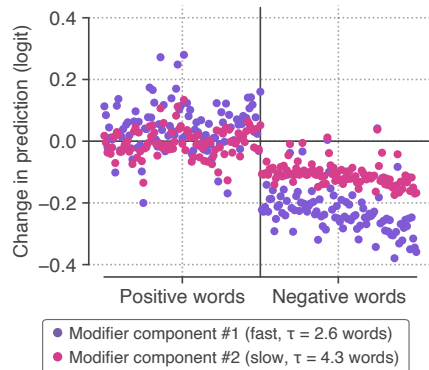

Figure 8. Barcodes for the principal components (eigenvectors) of the modification subspace. These barcodes show how the two dimensions of the modification subspace support rich modification of valence words with two time constants.

5.3.3. SUMMARY OF MODIFIER BARCODES

Beyond dynamics, we were also interested in how the input Jacobian changes in this two dimensional subspace. In particular, we wanted to understand whether or not there are different types of modification effects. To look at this,

we computed barcodes for the top two modifier axes[6] (Fig. 8). We saw that qualitatively, the two dimensions have similar effects, so the predominant difference between them is presumably the difference in timescales discussed above.

However, there are still rich patterns of effects in terms of the barcode weights for individual valence words. For example, the barcode weight corresponding to the valence word "stars" has a strong projection on the first modifier component, but not the second. This suggests that modifier words that induce important contextual changes for "stars" (e.g. "zero stars" and "five stars") should preferentially have larger projections onto the first modifier component, and not the second. Indeed, we see that "zero" and "five" are included in the set of modifier words, and have projections only on the first, faster modifier component (Fig. 6). By tuning the input Jacobian for different words along these two modifier components, the network is able to correspondingly tune the timescale of their effects.

5.3.4. PERTURBATION EXPERIMENT

Finally, to test whether the modifier subspace is necessary for contextual computation, we performed a perturbation experiment. To do this, we evaluated the network as usual except at every timestep we projected the hidden state out of the modifier subspace. Doing this, we reasoned, should interrupt contextual processing, but leave integration of valence intact. Indeed, we find that this perturbation has these expected effects both on single examples and across all test examples (Supp. Fig. 18, §F).

6. Other contextual effects

In addition to modifiers, we found contextual effects that are active during network processing of the beginning and end of documents.

6.1. Beginning of documents

After training, we found that the RNN's initial state, $\mathbf{h_0}$ (a vector of trainable parameters) is far from the line attractor (Fig. 1a), which surprised us. The reason for this was revealed when we computed the modifier barcode corresponding to the learned initial state (Fig. 9a). This barcode has a signature very much like that of an intensifier word such as "extremely", in that it accentuates both positive and negative valence. Moreover, we can also see that $\mathbf{h_0}$, when projected into the modifier subspace, has a significant projection on both modifier components (Fig. 9b). The implication of this is that words at the beginning of a document will be emphasized relative to words later in the document.

---

[6]Although we focus on the top two components in the main text, the barcodes for the top 10 modifier components are presented in Supp. Fig. 16.
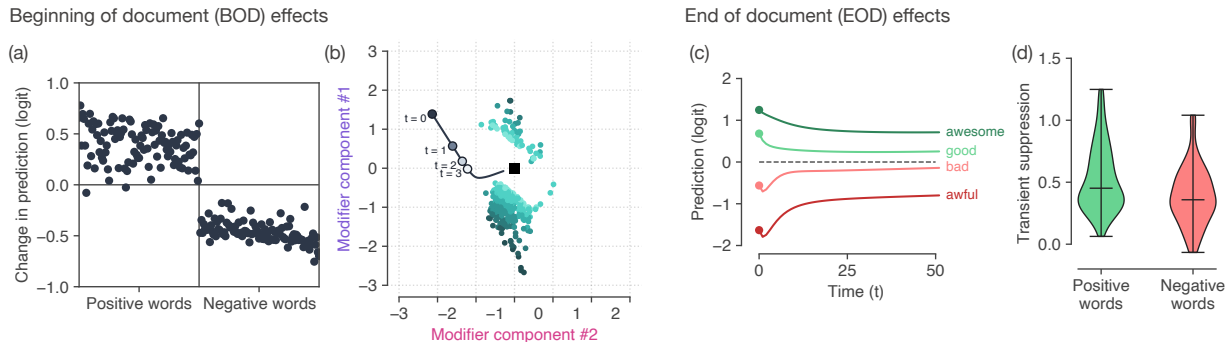
Figure 9. Contextual effects that occur at the beginning (a-b) and end (c-d) of documents. (a) Barcode measured at the initial state, $\mathbf{h_0}$, reveals that the initial state emphasizes sentiment. (b) Projection of the (trained) initial state (gray circle, $t = 0$) onto the modifier subspace and corresponding dynamics in response to 50 pad tokens (gray line, $t = 1, \ldots, 50$). The initial state initially has a large projection onto both modifier components, which means that words in the beginning of each document are *emphasized* (weighted more strongly) relative to words in the middle of the document. (c) When a valence word enters the network it introduces a transient deflection on the readout that decays in ~ 50 steps. This means that words at the end of a review are modified, as the transient has not had time to decay. In the four examples shown, the effect is to emphasize these words. (d) Summary of the amount of transient suppression across many words.

Presumably, the RNN learned to do this because it improves training performance. We tested this hypothesis directly using a perturbation experiment. We projected $\mathbf{h_0}$ out of the 2D modifier subspace, and then re-tested the GRU using this new initial state. We found that this perturbation caused a drop in accuracy of 0.17% on the test set and 0.23% on the train set, corresponding to an decrease in 68 and 1150 correct reviews, respectively. Importantly, projecting out a *random* 2D subspace did not affect the accuracy, indicating that this accuracy effect is unique to the modifier subspace.

6.2. End of documents

We also found contextual effects that emphasized words at the end of documents. Here, we identified a different mechanism responsible: short-term transient dynamics. When we looked at the impulse response of the system (Fig. 9c), we found that the projection onto the readout in response these tokens is initially large, but then decays over the course of around 50 tokens, until it settles to the steady-state valence of each word. We quantified this effect by computing the ratio of the steady-state valence to its initial valence. We did this for a large set of positive and negative words (Fig. 9d), and find consistent effects. The implication is that words at the end of a review are also emphasized, as their transient effects do not have time to decay away. We further identified two linear modes of the recurrent dynamics responsible for these transient effects (analyzed in Supp. Mat. §E).

We also tested that end of document emphasis matters to performance with another perturbation experiment. We added 50 pad tokens to the end of each review, thus allowing the transient dynamics to decay and thereby removing their effects along the readout. We evaluated the RNN on this padded dataset and found that the test accuracy dropped by

0.05% and the train accuracy by 0.2%, corresponding to incorrectly categorizing 20 and 1000 reviews, respectively.

7. Additional RNN architectures

So far, we have analyzed a particular RNN architecture, a GRU. We additionally trained and analyzed other RNN architectures, including an LSTM (Hochreiter & Schmidhuber, 1997), Update Gate RNN (Collins et al., 2016), and a Vanilla RNN. We found that all of the gated architectures were capable of processing modifiers. In particular, shuffling inputs reduced accuracy for the gated architectures by a similar amount (Figure 10a), and they have similar responses to test phrases with modifier words (Figure 10b). Finally, we repeated the analyses in Figures 3-6 for the LSTM, which are shown in Supp. Mat. §C. This hints at universal mechanisms underlying contextual processing in gated RNNs (Maheswaranathan et al., 2019a).

8. Bilinear model captures modifier effects

Taken together, our analyses reveals the mechanisms by which RNNs perform contextual processing. In particular, the key mechanism involves changing the input Jacobian ($\mathbf{J}^{\text{inp}}$) in the modifier subspace, and modifier words explicitly project the state into that subspace.

We can synthesize *how* the input Jacobian changes by expressing it as the sum of the input Jacobian at the line attractor, $\mathbf{J}^{\text{inp}}|_{(\mathbf{h}^*,\mathbf{0})}$, with P additional bilinear terms:

$$\mathbf{J}^{\text{inp}}|_{(\mathbf{h}^{\text{mod}},\mathbf{0})} = \mathbf{J}^{\text{inp}}|_{(\mathbf{h}^*,\mathbf{0})} + \sum_{p=1}^{P} \left((\mathbf{h}^{\text{mod}} - \mathbf{h}^*)^T \mathbf{m}^p \right) \mathbf{A}^p,$$
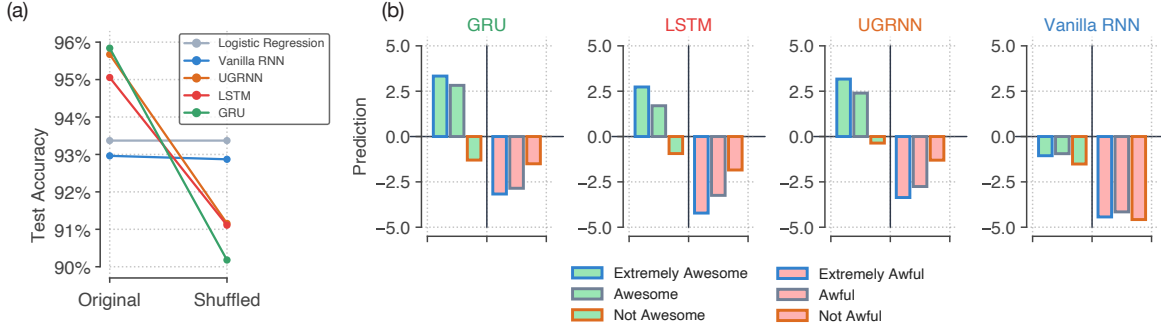
(6)

Figure 10. Summary across different architectures. (a) We performed a control experiment by randomly shuffling the tokens of each example sequence. We then classified these shuffled reviews using the RNNs trained on the original (non-shuffled) data. These results show that word order matters for gated RNNs but not for the Vanilla RNN or BoW model; thus the gated RNNs take word order into account. (b) Example predictions for RNNs for the phrases "**extremely** awesome", "awesome", "**not** awesome", "**extremely** awful", "awful", and "**not** awful". The gated RNN architectures show similar, modulated responses to these tokens while the Vanilla RNN shows little modulation of the valences of either "awesome" or "awful", regardless of the preceding modifier word.

where $\mathbf{A}^p$ is a fixed matrix (with the same dimensions as the input Jacobian) and $\mathbf{m}^p$ is a modifier component. Thus we capture the variation in $\mathbf{J}^{\text{inp}}$ as a function of $\mathbf{h}^{\text{mod}}$ by starting with the input Jacobian at $\mathbf{h}^*$, a point along the line attractor, and adding a small number of bilinear terms to it.

The additional P terms consist of two components. The first, $(\mathbf{h}^{\text{mod}}-\mathbf{h}^*)^T\mathbf{m}^p$, a scalar, captures the strength or amount of modification as a function of the projection of the hidden state onto the modifier component $\mathbf{m}^p$. As the hidden state decays back to the line attractor, the modification effects naturally decay along with the dot product $(\mathbf{h}_t-\mathbf{h}^*)^T\mathbf{m}^p$, with $\mathbf{h}_0=\mathbf{h}^{\text{mod}}$. The second term, $\mathbf{A}^p$, models the corresponding change in the input Jacobian. These $\mathbf{A}^p$ terms capture variation in the barcodes in Figure 8. We fit the parameters ($\mathbf{m}^p$ and $\mathbf{A}^p$) by regressing eq. (6) against the actual input Jacobian for the set of modifier words[7]. This approximation captures 83%, 96%, and 98% of the input Jacobian variation across all modification tokens for $P \in [1, 2, 3]$, respectively.

9. Augmented baseline models with insights from RNNs recover performance

The result that the bilinear model defined in equation (6) achieves 98% of the variance in $\mathbf{J}^{\text{inp}}|_{(\mathbf{h}^{\text{mod}},\mathbf{0})}$ with only $P = 3$ terms motivates baseline models of the form

$$\mathbf{w}^T\mathbf{J}^{\text{inp}}|_{(\mathbf{h}^*,\mathbf{0})} + \sum_{p=1}^{P}\left((\mathbf{h}^{\text{mod}}-\mathbf{h}^*)^T\mathbf{m}^p\right)\mathbf{w}^T\mathbf{A}^p + b, \quad (7)$$

where $\mathbf{w}$ and b are the readout weights and bias, respectively. First, we note that the first term $\mathbf{w}^T\mathbf{J}^{\text{inp}}|_{(\mathbf{h}^*,\mathbf{0})} \in \mathbb{R}^W$ is analogous to the weights $\boldsymbol{\beta} \in \mathbb{R}^W$ in a Bag-of-Words

[7]This regression can be solved in closed form via a singular value decomposition (SVD).

model (BoW). The additional P terms can be interpreted as additional $\boldsymbol{\beta}_{\text{mod}}$ weights that activate after a modifier word occurs and whose effect decays as $\mathbf{h}^{\text{mod}}$ returns to the line attractor. Thus, we propose augmented models of the form

$$\sum_{t=1}^{T}\left(\beta[t] + \sum_{p=1}^{P}(f^m * \mu^m)[t]\,\beta_{\text{mod}}^p[t]\right) + b, \quad (8)$$

where we have transitioned to indexing $\boldsymbol{\beta}$ by time t, which selects the weight for the word occurring at index t in the review. Modifier weights $\beta_{\text{mod}}^p[t]$ are defined separately from the baseline valence weights $\beta[t]$.

We model the decaying effect of modifiers via convolution of an indicator that signals the location of each modifier word, $\mu^m[t]$, with a causal exponential filter $f^m[s] = \alpha^m\exp(-s/\tau^m)$, with scaling α^m and timescale τ^m.

All of the augmented baseline models have the form given by equation (8), with different choices for what modifiers to include. In particular, we trained the following models (full descriptions are in Supp. Mat. §G):

Bag-of-Words (BoW): Baseline model with no modifiers.
Convolution of Modifier Words (CoMW): BoW plus convolutional modifiers with the *same* regression weights (modifiers do not have unique weights).
Convolution of BOD/EOD: BoW plus two additional modifier tokens for the beginning and end of the document (BOD/EOD), with the same regression weights.
CoMW + β_{mod}: BoW plus conv. modifiers with *learned* regression weights (modifiers have unique weights).
Convolution of BOD/EOD + $\beta_{\text{BOD}} + \beta_{\text{EOD}}$: BoW plus BOD/EOD modifiers, with learned regression weights.
CoMW + $\beta_{\text{mod}} + \beta_{\text{BOD}} + \beta_{\text{EOD}}$: BoW plus convolutional *and* BOD/EOD modifiers, with learned regression weights (the most powerful of the above models).

For all of our baseline experiments we set the number of modifiers, $M = 400$, and the number of modifier β_{mod} vectors to $P = 3$ (as $P = 3$ explains 98% of the variance of $\mathbf{J}^{\mathrm{inp}}|_{(\mathbf{h}^{\mathrm{mod}}, \mathbf{0})}$ in the bilinear model in §8). The models were trained using the Adam optimizer (Kingma & Ba, 2014). We selected hyperparameters (learning rate, learning rate decay, momentum, an ℓ_2 regularization penalty, and dropout rate) via a validation set using random search. We found dropout directly on the input words to be a useful regularizer for the more powerful models.

Results for these augmented baselines are in Table 1. The classic BoW model achieved 93.57% (on the Yelp dataset) and represents a baseline that does *not* implement contextual processing. The additional five baseline models range in increasing modeling power, with the most powerful baseline model achieving 95.63%, a test accuracy that is very close to the best performing RNN (90% of the difference).

Table 1. Test accuracies across baseline models and RNNs for the Yelp and IMDB datsets.

	YELP 2015	IMDB
BASELINE MODELS		
BAG OF WORDS	93.57%	89.47%
CONVOLUTION OF BOD & EOD TOKENS	93.99%	89.57%
CONV. + BOD & EOD TOKENS + $\beta_{\mathrm{BOD}} + \beta_{\mathrm{EOD}}$	94.37%	89.60%
CONVOLUTION OF MODIFIER WORDS (COMW)	94.76%	89.90%
COMW + β_{MOD}	95.40%	90.68%
COMW + $\beta_{\mathrm{MOD}} + \beta_{\mathrm{BOD}} + \beta_{\mathrm{EOD}}$	95.63%	90.75%
RNN MODELS		
GRU (CHO ET AL., 2014)	95.84%	89.63%
LSTM (HOCHREITER & SCHMIDHUBER, 1997)	95.05%	91.59%
UPDATE GATE RNN (COLLINS ET AL., 2016)	95.67%	89.43%
VANILLA RNN	92.96%	89.55%

10. Related Work

Understanding context has long been an important challenge in natural language processing (Morante & Sporleder, 2012; Mohammad, 2017). In particular, including pairs of words (bigrams) as features in simple BoW models significantly improves classification performance (Wang & Manning, 2012). Polanyi & Zaenen (2006) introduced the idea of "contextual valence shifters", which model contextual effects by shifting valence for a hand crafted set of modifier words. Further work refined these ideas, demonstrating their usefulness in improving sentiment classification accuracy (Kennedy & Inkpen, 2006; Taboada et al., 2011); identifying the amount and scope of the shifts combining human annotators (Ruppenhofer et al., 2015; Schulder et al., 2017; Kiritchenko & Mohammad, 2017) and automated methods (Choi & Cardie, 2008; Ikeda et al., 2008; Liu & Seneff, 2009; Li et al., 2010; Boubel et al., 2013); and finally using these lexicons to regularize RNNs (Teng et al., 2016; Qian et al., 2016). More recently, due to larger and more readily available datasets, neural networks achieve state-of-

the-art performance on sentiment classification (Dieng et al., 2016; Johnson & Zhang, 2016; Zhang et al., 2018), *without* explicitly building in contextual effects, often fine tuning larger models trained using unsupervised methods (Howard & Ruder, 2018; Sun et al., 2019; Yang et al., 2019).

Methods for interpreting and understanding the computations performed by recurrent networks include: inspecting individual units (Karpathy et al., 2015); visualizing salient inputs (Li et al., 2015; Ribeiro et al., 2016; Arras et al., 2017; Murdoch et al., 2018), analyzing and clustering RNN dynamics (Elman, 1990; 1991; Strobelt et al., 2016; Ming et al., 2017; Maheswaranathan et al., 2019b); and studying changes due to perturbed or dropped inputs (Kádár et al., 2017). For a review of these methods, and others, see Belinkov & Glass (2019).

11. Discussion

In summary, we provide tools and analyses that elucidate how neural networks implement contextual processing.

Here, we focused on unidirectional recurrent networks. Extending these tools to other tasks and architectures, such as those that use attention (Bahdanau et al., 2014; Vaswani et al., 2017), is a promising future research direction. For example, we predict that the reverse direction of a bidirectional RNN (Schulder et al., 2017) would reveal backwards modifiers (e.g. the suffix *-less* which has a suppressive effect on the preceding word stem, as in *soulless*).

Our analysis reveals rich modifier effects and timescales learned by the RNN. These properties are remarkably consistent with properties of modifiers from the linguistics literature, including: the length or scope of contextual effects lasting a few words (Chapman et al., 2001; Taboada et al., 2011; Reitan et al., 2015; Strohm & Klinger, 2018), the asymmetry in the strength and number of negators vs intensifiers (Horn, 1989; Kennedy & Inkpen, 2006; Taboada et al., 2011; Schulder et al., 2017), the relative weighting of different intensifiers (Ruppenhofer et al., 2015), and the fact that negation is better modeled as an additive effect rather than a multiplicative effect (Zhu et al., 2014).

This speaks to the general scientific program of analyzing optimized universal dynamical systems to provide insights into the underlying structure of natural data.

Acknowledgements

References

Arras, L., Montavon, G., Müller, K.-R., and Samek, W. Explaining recurrent neural network predictions in sentiment analysis. *arXiv preprint arXiv:1706.07206*, 2017.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Belinkov, Y. and Glass, J. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72, 2019.

Boubel, N., François, T., and Naets, H. Automatic extraction of contextual valence shifters. In *Proceedings of the international conference recent advances in natural language processing RANLP 2013*, pp. 98–104, 2013.

Casey, M. The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural computation*, 8(6):1135–1178, 1996.

Castelvecchi, D. Can we open the black box of AI? *Nature News*, 538(7623):20, 2016.

Chapman, W. W., Bridewell, W., Hanbury, P., Cooper, G. F., and Buchanan, B. G. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of biomedical informatics*, 34(5):301–310, 2001.

Cho, K., Merrienboer, B. v., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proc. Conference on Empirical Methods in Natural Language Processing*, Unknown, Unknown Region, 2014.

Choi, Y. and Cardie, C. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pp. 793–801, 2008.

Collins, J., Sohl-Dickstein, J., and Sussillo, D. Capacity and trainability in recurrent neural networks. *arXiv preprint arXiv:1611.09913*, 2016.

Cueva, C. J. and Wei, X.-X. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770*, 2018.

Dieng, A. B., Wang, C., Gao, J., and Paisley, J. TopicRNN: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*, 2016.

Elman, J. L. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Elman, J. L. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225, 1991.

Golub, M. and Sussillo, D. FixedPointFinder: A tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks. *Journal of Open Source Software*, 3(31):1003, 2018. doi: 10.21105/joss.01003.

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

Horn, L. *A natural history of negation*. University of Chicago Press, 1989.

Horn, L. R. and Kato, Y. *Negation and polarity: Syntactic and semantic perspectives*. OUP Oxford, 2000.

Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

Ikeda, D., Takamura, H., Ratinov, L., and Okumura, M. Learning to shift the polarity of words for sentiment classification. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, 2008.

Johnson, R. and Zhang, T. Supervised and semi-supervised text categorization using lstm for region embeddings. *arXiv preprint arXiv:1602.02373*, 2016.

Jordan, I. D., Sokol, P. A., and Park, I. M. Gated recurrent units viewed through the lens of continuous time dynamical systems. *arXiv preprint arXiv:1906.01005*, 2019.

Kádár, A., Chrupała, G., and Alishahi, A. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43(4):761–780, 2017.

Karpathy, A., Johnson, J., and Fei-Fei, L. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Kennedy, A. and Inkpen, D. Sentiment classification of movie reviews using contextual valence shifters. *Computational intelligence*, 22(2):110–125, 2006.

Khalil, H. K. *Nonlinear Systems*. Pearson, 2001. ISBN 0130673897.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kiritchenko, S. and Mohammad, S. M. The effect of negators, modals, and degree adverbs on sentiment composition. *arXiv preprint arXiv:1712.01794*, 2017.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Li, J., Chen, X., Hovy, E., and Jurafsky, D. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.

Li, S., Lee, S. Y. M., Chen, Y., Huang, C.-R., and Zhou, G. Sentiment classification and polarity shifting. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 635–643. Association for Computational Linguistics, 2010.

Liu, J. and Seneff, S. Review sentiment scoring via a parse-and-paraphrase paradigm. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pp. 161–169. Association for Computational Linguistics, 2009.

Maheswaranathan, N., Williams, A., Golub, M., Ganguli, S., and Sussillo, D. Universality and individuality in neural dynamics across large populations of recurrent networks. In *Advances in Neural Information Processing Systems 32*, pp. 15603–15615. Curran Associates, Inc., 2019a.

Maheswaranathan, N., Williams, A., Golub, M., Ganguli, S., and Sussillo, D. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019b.

Mante, V., Sussillo, D., Shenoy, K. V., and Newsome, W. T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78, 2013.

Ming, Y., Cao, S., Zhang, R., Li, Z., Chen, Y., Song, Y., and Qu, H. Understanding hidden memories of recurrent neural networks. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 13–24. IEEE, 2017.

Mohammad, S. M. Challenges in sentiment analysis. In *A practical guide to sentiment analysis*, pp. 61–83. Springer, 2017.

Morante, R. and Sporleder, C. Modality and negation: An introduction to the special issue. *Computational linguistics*, 38(2):223–260, 2012.

Murdoch, W. J., Liu, P. J., and Yu, B. Beyond word importance: Contextual decomposition to extract interactions from LSTMs. In *International Conference on Learning Representations*, 2018.

Polanyi, L. and Zaenen, A. Contextual valence shifters. In *Computing attitude and affect in text: Theory and applications*, pp. 1–10. Springer, 2006.

Qian, Q., Huang, M., Lei, J., and Zhu, X. Linguistically regularized lstms for sentiment classification. *arXiv preprint arXiv:1611.03949*, 2016.

Quirk, R., Greenbaum, S., Leech, G., and Svartvik, J. *A Comprehensive Grammar of the English Language*. Longman, 1985.

Reitan, J., Faret, J., Gambäck, B., and Bungum, L. Negation scope detection for twitter sentiment analysis. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 99–108, 2015.

Ribeiro, M. T., Singh, S., and Guestrin, C. Why should I trust you? explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

Rodriguez, P., Wiles, J., and Elman, J. L. A recurrent neural network that learns to count. *Connection Science*, 11(1): 5–40, 1999.

Ruppenhofer, J., Brandes, J., Steiner, P., and Wiegand, M. Ordering adverbs by their scaling effect on adjective intensity. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pp. 545–554, 2015.

Schulder, M., Wiegand, M., Ruppenhofer, J., and Roth, B. Towards bootstrapping a polarity shifter lexicon using linguistic features. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 624–633, 2017.

Seung, H. S. How the brain keeps the eyes still. *Proceedings of the National Academy of Sciences*, 93(23): 13339–13344, 1996. ISSN 0027-8424. doi: 10.1073/pnas.93.23.13339.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Strobelt, H., Gehrmann, S., Huber, B., Pfister, H., Rush, A. M., et al. Visual analysis of hidden state dynamics in recurrent neural networks. *CoRR, abs/1606.07461*, 2016.

Strohm, F. and Klinger, R. An empirical analysis of the role of amplifiers, downtoners, and negations in emotion classification in microblogs. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 673–681. IEEE, 2018.

Sun, C., Qiu, X., Xu, Y., and Huang, X. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pp. 194–206. Springer, 2019.

Sussillo, D. and Barak, O. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural computation*, 25(3):626–649, 2013.

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., and Stede, M. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, 2011.

Teng, Z., Vo, D.-T., and Zhang, Y. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1629–1638, 2016.

Tsung, F.-S. and Cottrell, G. W. Phase-space learning. In *Advances in Neural Information Processing Systems*, pp. 481–488, 1995.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, S. and Manning, C. D. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*, pp. 90–94. Association for Computational Linguistics, 2012.

Wiegand, M., Balahur, A., Roth, B., Klakow, D., and Montoyo, A. A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, NeSp-NLP '10, pp. 60–68, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1858959.1858970.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pp. 5754–5764, 2019.

Zhang, L., Wang, S., and Liu, B. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4): e1253, 2018. doi: 10.1002/widm.1253.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 649–657. Curran Associates, Inc., 2015.

Zhu, X., Guo, H., Mohammad, S., and Kiritchenko, S. An empirical study on the effect of negation words on sentiment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 304–313, 2014.

Zipser, D. Recurrent network model of the neural mechanism of short-term active memory. *Neural Computation*, 3(2):179–193, 1991.