A. Toy sentiment task

A.1. Task introduction

In order to illuminate how negation or emphasis is implemented in RNNs we developed a small toy language to isolate their 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", "awesome"). In addition, we added two modifier words, an intensifier that doubled the valence of the next input (analgous to words such as "**extremely**"), and a negator that reversed the valence of the next *four* inputs (analogous to words such as "**not**"). We varied the timescales of the modifier effects because there was evidence for this in the Yelp 2015 sentiment data we analyzed and also because we were interested in understanding modifier words with different dynamics.

We generated random reviews using this language by randomly selecting 50 words and placing them sequentially. To avoid overly complex effects we made sure that "not" could not follow "not" within 4 tokens and that "extremely" could not directly follow "extremely". We trained RNNs to track the corresponding sentiment, defined as the per-time step cumulative sum of potentially modified valence inputs. In this way, we could explicitly define per-time step target values for the output, thus controlling how an intensifier or negator works. As we specified a per-time step integration target for the network, we used a least-mean-squares loss, as opposed to cross-entropy loss typically used for classification. These are differences from the standard sentiment classification setup where a review is only associated with a single positive or negative classification pertaining to the entire review. For example, after training, the RNN correctly integrated long sequences of words and we verified that the RNN integrated "good" as +1, "**extremely** good" as +2 and "**not** the the the good" as -1.

A.2. Confirmation of line attractor dynamics in the toy model

We analyzed the trained networks using the methods developed by Maheswaranathan et al. (2019b). An example state-space plot from a trained network is shown in Figure 2. We verified that the network indeed has line attractor dynamics, as in (Maheswaranathan et al., 2019b). More precisely, using fixed point optimization (see Supplemental Methods Section B) we identified a 1D manifold of slow points as shown in Figure 2. Each one of these slow points was approximately marginally stable, meaning in this case that a single mode of the linearized system was marginally stable. This can be seen in the eigenvalue plot in Supp. Figure 11. The eigenvalue associated with this mode is very close to $(1, 0)$ in the complex plane.

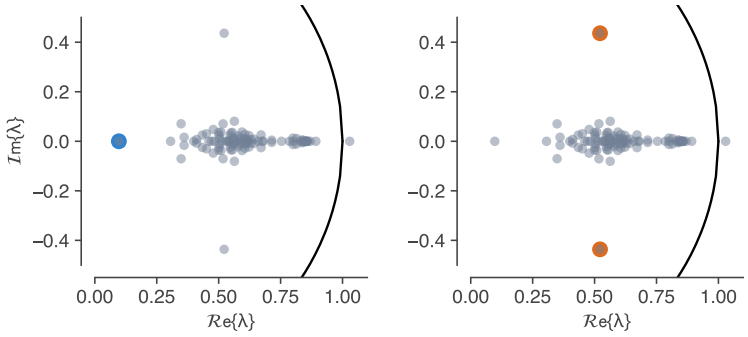

Figure 11. GRU trained on toy sentiment task. Shown are the eigenvalue plots for two fixed points along the line attractor. These fixed points are the ones closest to the $\mathbf{h}^{\text{extremely}}$ and $\mathbf{h}^{\text{not}}$ operating points, on the left and right panels, respectively. In both plots there is an integration mode near $(1, 0)$. In the left panel, the eigenvalue associated with the mode for the single-step decay dynamic for "**extremely**" is shown in blue. In the right panel, the eigenvalues associated with the decay dynamic for the "**not**" token are shown in orange. The "**extremely**" emphasis was defined for only the next word, while the "**not**" negation was defined for four time steps.

A.3. Analysis of state deflections caused by inputs after modifier tokens

The first analysis we performed beyond those done in (Maheswaranathan et al., 2019b) was to examine whether effects of modifier tokens could be well-approximated by a carefully chosen linearization. We chose

$$\mathbf{h}_t \approx F(\mathbf{h}^{\text{mod}}, \mathbf{0}) + \mathbf{J}^{\text{inp}}|_{(\mathbf{h}^{\text{mod}}, \mathbf{0})} \mathbf{x}_t, \tag{9}$$

with $\mathbf{h}^{\text{mod}}$ denoting the state immediately after a modifier word enters the system. We measured the quality of equation (9) by computing the output value

$$\mathbf{w}^T \mathbf{J}^{\text{inp}}|_{(\mathbf{h}^e,\mathbf{0})}\mathbf{x}_t + b, \tag{10}$$

for all valence words when $\mathbf{h}^e$ was set to be either 1) the state somewhere along the line attractor (denoted $\mathbf{h}^*$), which measures normal integration dynamics, or 2) the state after a modifier word (denoted $\mathbf{h}^{\text{mod}}$), which measures modified integration dynamics, so $\mathbf{h}^e \in \{\mathbf{h}^*, \mathbf{h}^{\text{mod}}\}$. We show the results in Supp. Figure 12. The results show the linear expansion is very good, yielding a 92% accuracy in comparison to the full nonlinear update.

The quality of these expansion results now gives an abstract explanation for how the RNN implements context via modification words. A modifier enters the system and causes a deflection to $\mathbf{h}^{\text{mod}}$. The computational purpose for this deflection is to enter a region of state space that modifies processing of subsequent word(s). This can be measured by $\mathbf{J}^{\text{inp}}$, as in equation (9) and to good approximation is essentially a linear update to $\mathbf{h}^{\text{mod}}$, as measured by the effect on the readout.

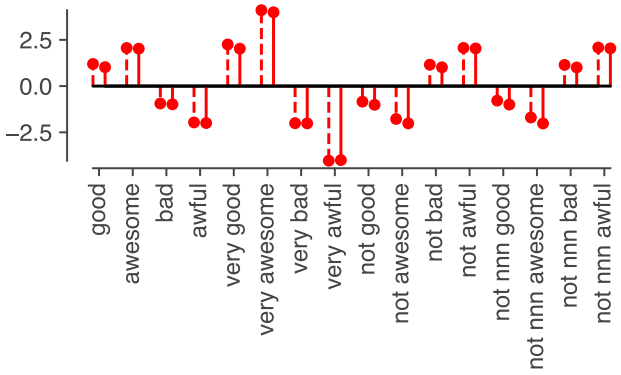

Figure 12. The effect of modifier tokens in the GRU trained on toy sentiment task. We computed $\mathbf{J}^{\text{inp}}|_{\mathbf{h}^e,\mathbf{0}}\ \mathbf{x}$ as described in the text for both modifier inputs and non-modifier inputs. We then measured the readout values for all valence tokens and modifiers (red dashed lines). We also measured the readout value for the full nonlinear update from $\mathbf{h}^e$ (red solid lines). The linear update on the readout is approximate to 91.8% as measured by the mean absolute value of the ratio of the approximation over the full nonlinear update. Note that "nnn" means three noise words between "not" and the valence word.

A.4. Analysis of state dynamics that participate in modifier computation

Having established how modifier words cause contextual state deflections we became interested in how the RNN implemented the transient response to each modifier. In particular, as we defined the effect of "extremely" to last one token and the effect of "not" to last for four, we expected there to be different length transients induced by these modifiers. To investigate this, we studied the linear system defined by

$$\Delta\mathbf{h}_t \approx \mathbf{J}^{\text{rec}}|_{(\mathbf{h}^*,\mathbf{x}^*)}\Delta\mathbf{h}_{t-1}, \tag{11}$$

where $\mathbf{x}^* = \mathbf{0}$, and $\mathbf{h}^*$ was a fixed point along the line attractor chosen because it was closest to $\mathbf{h}^{\text{mod}}$, where $\mathbf{h}^{\text{mod}} = F(\mathbf{h}_0, \mathbf{x}^{\text{mod}})$ and $\mathbf{h}_0$ is the system initial condition. Here $\mathbf{x}^{\text{mod}}$ ranges over "extremely" and "not".

Examination of the eigenvalue spectra for linearizations around the fixed points along the line attractor (Supp. Figure 11) revealed three obvious modes far away from the bulk of the eigenvalue distribution. We wondered whether any of these three modes were responsible for implementing the transient dynamics associated with the two modifier words. We reasoned that if these modes of were of any utility to modifier dynamics, then a projection of the state onto these modes should be apparent when a modifier word enters the RNN. Therefore, we computed the subspace angle between ($\mathbf{h}^{\text{mod}}$ - $\mathbf{h}^*$) and $\boldsymbol{\ell}_a$, the a^{th} left eigenvector of the recurrent Jacobian from equation (11). The results are shown in Supp. Fig. 13 and indeed show that the isolated extremely fast decaying mode is related to processing of "extremely" and that the oscillatory mode is related to processing the "not" token. It is the modes with smallest subspace angle that are highlighted with color in Supp. Fig. 11.

Finally, we reasoned that if these modes of the linearized systems were important for processing the effect of modifier inputs, then removing the projection onto those modes should remove the effect of the modifier. We therefore rank ordered the left eigenvectors by how much their individual removal perturbed the effect of the modifier. Specifically, we defined the state

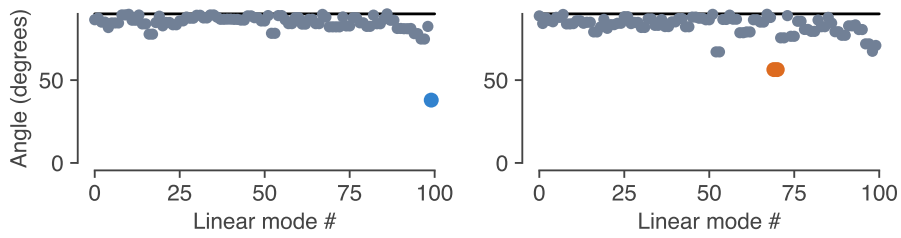

Figure 13. Subspace angles between state space deflections caused by modifier tokens and the left eigenvectors of the linearized systems around fixed points. Left panel. The subspace angles (degrees) between $(\mathbf{h}^{\text{extremely}} - \mathbf{h}^*)$ and $\boldsymbol{\ell}_a$ for the a^{th} left eigenvector of the linear system around the fixed point closest to $\mathbf{h}^{\text{extremely}}$. Right panel is the same as the left panel, except for analysis of the "not" modifier.

deflection of a valence word after a modifier word as $\mathbf{h}^{\text{valence}} = F(\mathbf{h}^{\text{mod}}, \mathbf{x}^{\text{valence}})$, then we measured error in $\mathbf{w}^T\mathbf{h}^{\text{valence}} + \mathbf{b}$ as the components of $\mathbf{h}^{\text{mod}}$ that projected onto $\boldsymbol{\ell}_a$ were removed, for all a. We then computed the cumulative sum of these effects, starting with the removal of the mode that was most important and then removing both the first and second most important modes, etc.[8] The results are shown in Supp. Fig. 14 and confirm that removing even the single mode with largest projection into $(\mathbf{h}^{\text{mod}} - \mathbf{h}^*)$ essentially completely destroys the modification effect.

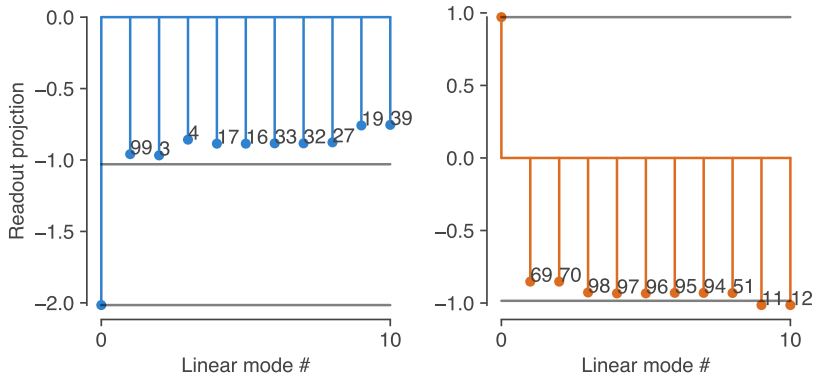

Figure 14. Removing the components of $\mathbf{h}^{\text{mod}}$ that project onto select left eigenvectors of the linearized systems around fixed points completely destroys the effect of the modifier input. See text for methodological details. Left) The result of removing $\boldsymbol{\ell}_{99}$, a very fast decaying mode, completely destroyed the effect of the "extremely" modifier token on the valence word "bad". Right) removing the modes $\boldsymbol{\ell}_{69}$ and $\boldsymbol{\ell}_{70}$ completely destroyed the effect of the "not" modifier token on the valence word "bad". In this case there were three noise words used, e.g. "not the the the bad".

B. Fixed point finding methodology

We treat any recurrent neural network (RNN) update as a function F, that updates a hidden state $\mathbf{h}$ in response to some input: $\mathbf{h}_t = F(\mathbf{h}_{t-1}, \mathbf{x}_t)$. This defines a discrete time dynamical system. *Fixed points* of these dynamics are given by points $\mathbf{h}^*$ where applying the function F does not change the state: $\mathbf{h}^* = F(\mathbf{h}^*, \mathbf{x})$, for some input $\mathbf{x}$. Here, we focus on fixed points in the absence of input ($\mathbf{x} = \mathbf{0}$).

To computationally identify fixed points of the recurrent dynamics, we numerically solve the following optimization problem (Sussillo & Barak, 2013; Golub & Sussillo, 2018):

$$\min_{\mathbf{h}} \frac{1}{2}\|\mathbf{h} - F(\mathbf{h}, \mathbf{0})\|_2^2. \tag{12}$$

[8]For complex conjugate pairs, we always removed the pair when one vector was slated to be removed.

In general, the function minimized in equation (12) has many local minima, corresponding to different slow points or fixed points of F. We are interested in finding all slow points, regardless of whether they are local minima, such that $|\mathbf{h}^* - F(\mathbf{h}^*, \mathbf{0})|$ is significantly smaller than $|\mathbf{J}^{\text{rec}}|_{(\mathbf{h}^*,\mathbf{0})}\Delta\mathbf{h}_{t-1}|$ in $\mathbf{h}_t \approx F(\mathbf{h}^*, \mathbf{0}) + \mathbf{J}^{\text{rec}}|_{(\mathbf{h}^*,\mathbf{0})}\Delta\mathbf{h}_{t-1}$, thereby enabling high-quality linear approximations to the dynamics. We find these by running the optimization problem above starting from a large set of initial points, taken from the randomly selected RNN hidden states visited during test examples (Maheswaranathan et al., 2019b). We initialized the fixed point finding routine using 10,000 hidden states of the RNNs while performing sentiment classification.

C. Additional architectures

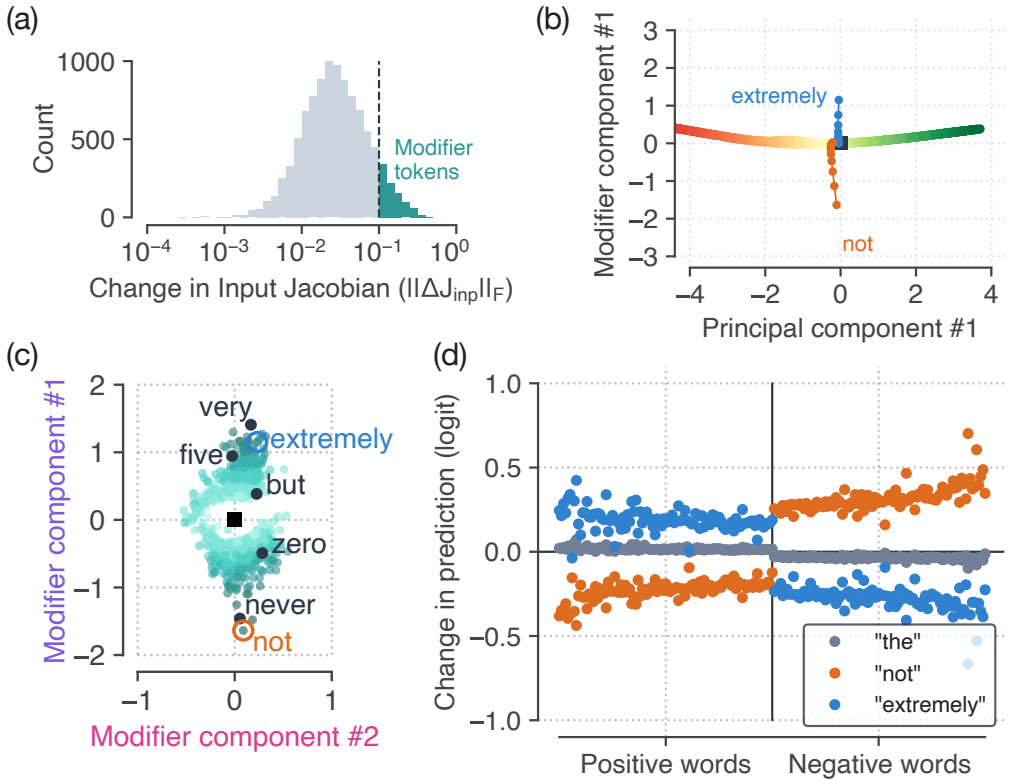

Figure 15. Analysis of modifier effects for an LSTM. This figure reproduces the analyses for an LSTM that are shown in the main paper for a GRU. It shows that LSTMs implement contextual effects of modifier words in a way very similar to the GRU. (a) Histogram of modifiers using data driven approach, as in Fig. 3. (b) Impulse response to modifier words, as in Fig. 4a. (c) Modifier subspace showing arrangement of various modifier words, as in Fig. 6. (d) Modifier barcodes for "**not**", "the", and "**extremely**" as in Fig. 5.

In the main text, we provided a detailed analysis of a particular RNN, a gated recurrent unit (GRU) (Cho et al., 2014). We were interested in whether our results were sensitive to this particular RNN architecture. To explore this, we trained an analyzed additional RNN architectures on the same task. Supp. Fig. 15 shows the results of applying our analyses to a trained LSTM (Hochreiter & Schmidhuber, 1997). We find a remarkable degree of similarity between the two networks (compare Fig. 3, Fig. 4a, Fig. 6, and Fig. 5 from the main text to Supp. Fig. 15a-d, respectively).

We see similar results for the other gated RNN architecture we trained, the Update Gate RNN (UGRNN; (Collins et al., 2016)). However, for the Vanilla RNN, we find that it does *not* have a clear modifier subspace, and is incapable of correctly processing contextual effects (Fig. 10). We suspect this is due to difficulties in training vanilla RNNs, rather than due to reduced modeling capacity, as the mechanisms proposed in this paper can in theory be implemented using Vanilla RNNs.

The degree of similarity in the mechanisms learned by gated RNNs for contextual processing suggest that these mechanisms may be universal computational motifs (Maheswaranathan et al., 2019a).

D. Additional barcodes

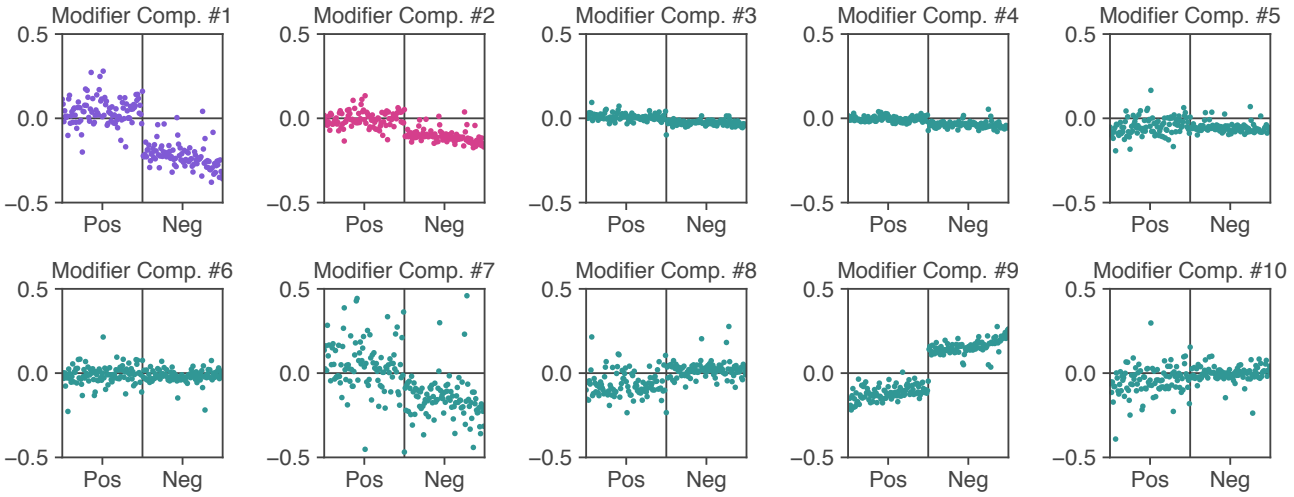

Figure 16. Additional modifier barcodes. For every modifier component, we compute a barcode to summarize the effect of that particular modifier dimension on inputs. The barcode summarizes the change in sensitivity to particular salient inputs (here, the top 100 positive and top 100 negative words). Barcode values of zero indicate that the sensitivity to those words is not affected (that is, there are no contextual effects). Each panel shows the barcode corresponding to a given modifier component. We additionally highlight the first (purple) and second (pink) components using different colors.

The barcodes presented in Figure 8 show the effects of modifiers along the top two modifier components (colored in purple and pink, respectively). Below (Fig. 16), we present additional barcodes for the top 10 modifier components. Note that nearly all of the variance in the modifier subspace is captured by just a few (2-3) components. Perturbation experiments also suggest that contributions from modifier components outside the top three to overall accuracy are minor (Supp. Fig. 18, §F). However, these additional barcodes do contain interesting structure. In particular, components seven and nine may have subtle contributions to processing of modifiers.

E. Transient EOD effects are implemented using two modes

We observed contextual processing at the end of a document, implemented using transient decay dynamics. That is, valence words would initially induce a large projection onto the readout, but this would decay to a steady-state value (Fig.9b). The steady-state effect occurs due to integration along the line attractor (Maheswaranathan et al., 2019b).

For the transient effects, we studied modes outside of the integration modes (those associated with slow eigenvalues) and modifier dynamics (those associated with very fast modes on the timescale of tens of tokens). In particular, we found two additional modes which were strongly activated by valence words, mode 78 (with a corresponding timescale of 19.57 tokens) and mode 194 (with a timescale of 6.63 tokens).

To quantify the transient suppression for these two modes, we computed the (instantaneous) change in prediction when we applied the linearized RNN with individual modes (eigenmodes) removed. This allows us to examine how important individual modes are to the instantaneous processing of valence tokens. Removing modes 78 and 194 resulted in strong changes in the valence associated with positive and negative words (Supp. Fig. 17a). To verify the timescale of these transient effects, we computed the average change in prediction across a set of 100 valence (50 positive and 50 negative) words over time. This allows us to investigate how individual eigenmodes contribute to the overall transient (Supp. Fig. 17b). Finally, we performed a perturbation experiment to examine the effect of these two modes in particular. We probed the system's step response to valence tokens ("awesome", "good", "bad", and "'awful"), and either ran the full system or projected the hidden state out of subspace defined by modes 78 and 194 (Fig. 17c). This perturbation reduced or eliminated the transient effects.

Taken together, these results suggest that the RNN implements end of document contextual processing using two additional modes of the recurrent dynamics. These modes induce a transient that decays with timescales of around 7 and 20 tokens.

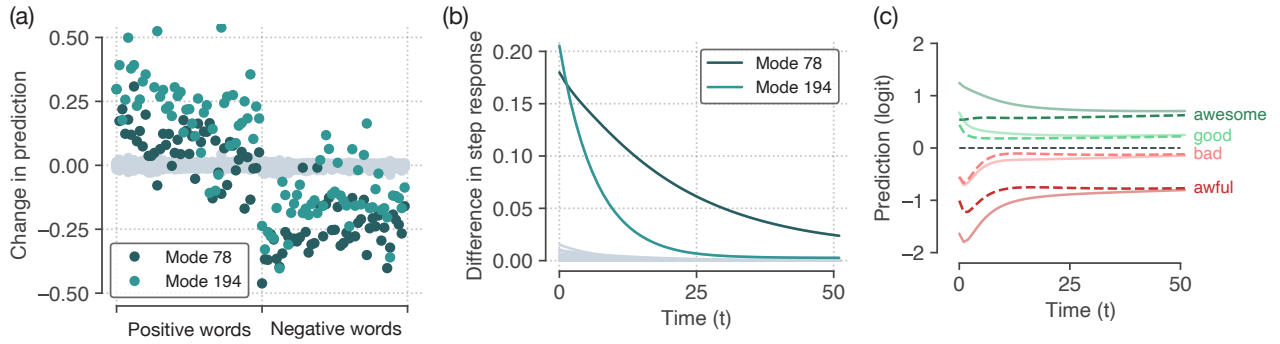

Figure 17. An analysis of transient linear dynamics that contribute to the end of document contextual effects. (a) We analyzed the linear modes of the recurrent Jacobian around a typical fixed point on the line attractor. In particular, we studied the projection of the linear modes onto the readout as a function of eigenvalue decay. We found two modes, #78, and #194 (dark green and light green respectively) that had substantial projections onto the readout and correspondingly large effects on the transient valence of positive and negative words (gray shows the change in prediction for the other modes). (b) The average absolute change in transient valence to valence words achieved by modes #78 and #194. (c) The transient valence of "awesome", "good", "bad", "awful" are reproduced from Figure 9 (solid lines), while the effect of removing the projection of the readout onto transient modes #78 and #194 are also shown (dashed lines).

Therefore, the last 20 or so tokens in a given document will be emphasized relative to those in the middle of a document.

F. Perturbations

Here, we perform perturbation experiments to eliminate particular mechanisms in the RNN. This demonstrates that these mechanisms are necessary for particular function. To test whether the modifier subspace is necessary for contextual processing, we probed the RNN with the same examples from Fig. 1a, but at every step we project the RNN hidden state *out* of either the modifier subspace (Supp. Fig. 18a) or a random control subspace (Supp. Fig. 18b). The modifier subspace perturbation selectively removes the network's ability to process contextual inputs. Finally, we ran the same perturbation for all of the examples in the test set. We found a negligible effect on accuracy when projecting out of a random subspace, but a significant reduction in accuracy when projecting out of the modifier subspace (Supp. Fig. 18c).

G. Augmented baseline models

Below, we provide full definitions for all of the augmented baseline models tested in §9.

Bag-of-words model ($W+1$ parameters): Each word in the document is associated with a scalar weight. These weights are then summed along with a bias.

$$\sum_{t=1}^{T} \beta[t] + b \tag{13}$$

Convolution of Modifier Words (CoMW) ($W+1+2M$ parameters): Modifier words scale the same weights used to estimate the valence of each word without modification. The modifier scaling decays exponentially with time.

$$\sum_{t=1}^{T} \left(\beta[t] + \sum_{m=1}^{M} \left(f^m * \mu^m\right)[t]\ \beta[t] \right) + b \tag{14}$$

Convolution of BOD and EOD Tokens ($W+1+4$ parameters): We augmented each example with Beginning of Document (BOD) token and End of Document (EOD) tokens. For the EOD token, the exponential filter was flipped acausally.

$$\sum_{t=1}^{T} \left(\beta[t] + \sum_{m \in [\text{BOD},\text{EOD}]} \left(f^m * \mu^m\right)[t]\ \beta[t] \right) + b \tag{15}$$

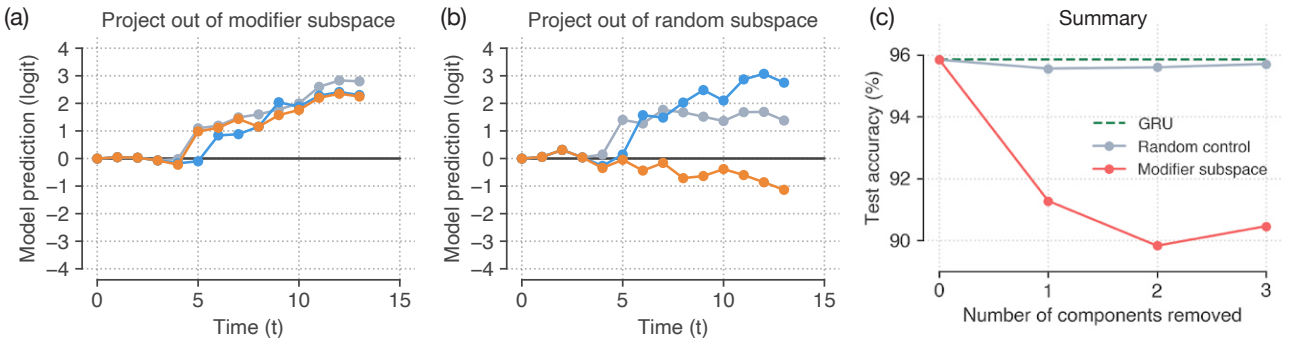

Figure 18. Perturbation experiment demonstrates that the modifier subspace is necessary for contextual processing. (a) Perturbation where a trained network is probed with three example sentences, "This movie is awesome. I like it." (gray), "This movie is **not** awesome, I **don't** like it." (orange), and "This movie is **extremely** awesome, I **definitely** like it." (blue), and where the hidden state is projected *out* of the modifier subspace (here, the modifier subspace is three dimensional) at each iteration. This perturbation collapses the trajectories, indicating that the system is no longer capable of processing modifier tokens. Moreover, the effect of this perturbation is to selectively remove contextual effects, but does not affect the baseline integration. (b) Random control where we project the hidden state out of a *random* three dimensional subspace. We see that this has no effect on the network activity (compare with Fig. 1b). (c) Summary across all test examples. Performance of original network (dashed green line) compared with perturbations where we project the hidden state at each timestep or iteration when applying the RNN. Projecting out of a random three dimensional subspace (solid gray line) has a negligible effect on the accuracy, whereas projecting out of the modifier subspace (solid red line) has a significant effect, equivalent to shuffling input tokens (Fig. 10a).

Convolution of Modifier Words + $\boldsymbol{\beta}_{\text{mod}}$ ($W + 1 + 2M + PW$ parameters): Modifier words exponentially scale additional learned weight vectors used exclusively to estimate modification.

$$\sum_{t=1}^{T}\left(\beta[t] + \sum_{p=1}^{P}\sum_{m=1}^{M}\left(f^m * \mu^m\right)[t]\ \beta^p_{\text{mod}}[t]\right) + b \tag{16}$$

Convolution of BOD and EOD Tokens + $\boldsymbol{\beta}_{\text{BOD}} + \boldsymbol{\beta}_{\text{EOD}}$ ($W + 1 + 4 + 2W$ parameters): Similar to Convolution of BOD and EOD Tokens, except that we allow the learning of two separate β_{mod} weights to estimate the effects of both the beginning and end of the review.

$$\sum_{t=1}^{T}\left(\beta[t] + \sum_{m\in[\text{BOD,EOD}]}\left(f^m * \mu^m\right)[t]\ \beta^m_{\text{mod}}[t]\right) + b \tag{17}$$

Convolution of Modifier Words + $\boldsymbol{\beta}_{\text{mod}} + \boldsymbol{\beta}_{\text{BOD}} + \boldsymbol{\beta}_{\text{EOD}}$ ($W + 1 + 4 + 2W + 2M + PW$ parameters): Combines the most powerful model versions from above to learn both modifier word effects as well as contextual effects at the beginning and end of the review.

$$\sum_{t=1}^{T}\left(\beta[t] + \sum_{m\in[\text{BOD,EOD}]}\left(f^m * \mu^m\right)[t]\ \beta^m_{\text{mod}}[t] + \sum_{p=1}^{P}\sum_{m=1}^{M}\left(f^m * \mu^m\right)[t]\ \beta^p_{\text{mod}}[t]\right) + b \tag{18}$$