Supplementary Information for: What made you do this? Understanding black-box decisions with sufficient input subsets

Contents

List of Figures

List of Tables

S1 Detailed Description of Alternative Methods

In Section 3, we describe a number of alternative methods for identifying rationales for comparison with our method. We use methods based on integrated gradients (Sundararajan et al., 2017), LIME (Ribeiro et al., 2016), and feature perturbation. Note that integrated gradients is an attribution method which assigns a numerical score to each input feature. LIME likewise assigns a weight to each feature using a local linear regression model for f around $\mathbf{x}$. In the perturbative approach, we compute the change in prediction when each feature is individually masked, as in Equation 1 (of Section S4.4). Each of these feature orderings R is used to construct a rationale using the **FindSIS** procedure (Section 3) for the "Suff. IG," "Suff. LIME," and "Suff. Perturb." (*sufficiency constrained*) methods.

Note that our text classification architecture (described in Section S4.2) encodes discrete words as 100-dimensional continuous word embeddings. The integrated gradients method returns attribution scores for each coordinate of each word embedding. For each word embedding $x_i \in \mathbf{x}$ (where each $x_i \in \mathbb{R}^{100}$), we summarize the attributions along the corresponding embedding into a single score y_i using the L_1 norm: $y_i = \sum_d |x_{id}|$ and compute the ordering R by sorting the y_i values.

We use an implementation of integrated gradients for Keras-based models from `https://github.com/hiranumn/IntegratedGradients`. In the case of the beer review dataset (Section 4.1), we use the mean embedding vector as a baseline for computing integrated gradients. In the case of TF binding (Section 4.2), we use the $[0.25, 0.25, 0.25, 0.25]$ uniform mean vector as the baseline reference value. As suggested in Sundararajan et al. (2017), we verified that the prediction at the baseline and the integrated gradients sum to approximately the prediction of the input.

For LIME and our beer reviews dataset, we use the approach described in Ribeiro et al. (2016) for textual data, where individual words are removed entirely from the input sequence. In our TF binding dataset, LIME replaces bases with the unknown `N` base (represented as the uniform-distribution $[0.25, 0.25, 0.25, 0.25]$). We use the implementation of LIME at: `https://github.com/marcotcr/lime`. The `LimeTextExplainer` module is used with default parameters, except we set the maximal number of features used in the regression to be the full input length so we can order all input features.

Additionally, we explore methods in which we use the same ordering R by these alternative methods but select the same number of input features in the rationale to be the median SIS length in the SIS-collection computed by our method on each example: the "IG," "LIME," and "Perturb." (*length constrained*) methods. In the TF binding models, we use a baseline of zero vectors such that the integrated gradients result along the encoded sequence is also one-hot. We compute the feature ordering based on the absolute value of the non-zero integrated gradient attributions.

In TF binding data (Section 4.2), we add an additional method, "Top IG," in which we compute integrated gradients using an all-zeros baseline and order features by attribution magnitude (as in the length constrained IG method). But, we select elements for the rationale by finding the minimum number of elements necessary such that the sum of integrated gradients of those features equals $\tau - f(\mathbf{0})$, where $\mathbf{0}$ is the all-zeros baseline for integrated gradients. Note that for the length constrained and Top IG methods, there is no guarantee of sufficiency $f(\mathbf{x}_S) \geqslant \tau$ for any input subset S.

S2 Details of the Transcription Factor Binding Analysis

S2.1 Dataset and Model

We use the *motif occupancy* datasets[1] from Zeng et al. (2016), where each dataset originates from a ChIP-seq experiment from the ENCODE project (Consortium et al., 2012). Each of the 422 datasets studies a particular transcription factor, containing between 600 and 700,000 (median 50,000) 101 base-pair DNA sequences (inputs) each associated with a binary label based on whether the sequence is bound by the TF or not. Each dataset also contains a test set ranging between 150 and 170,000 sequences (median 12,000). Here, the positive and negative classes in each dataset are balanced, and we filter out all sequences containing the unknown base (`N`).

[1]available at `http://cnn.csail.mit.edu`

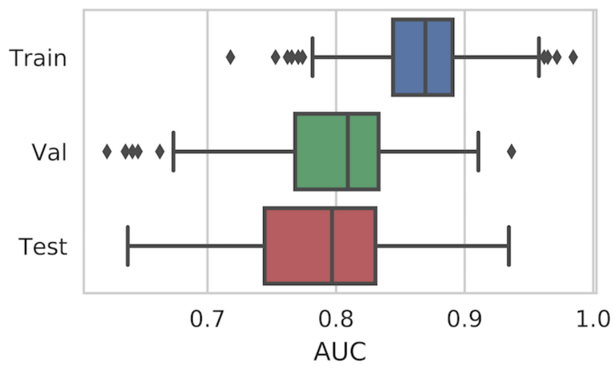


Figure S1: Median area under the receiver operating curve (AUC) for all 422 transcription factor binding motif occupancy datasets. The validation set is held-out at training but used to choose model parameters; the test set is not seen until after training.

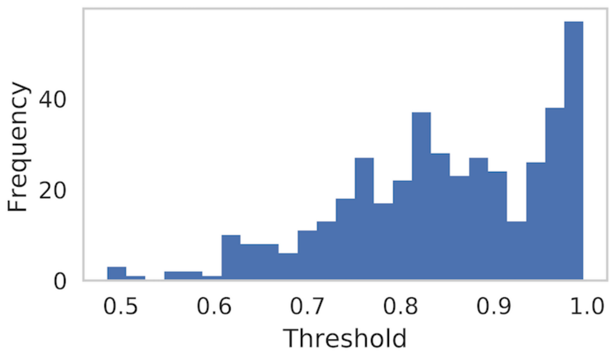


Figure S2: Thresholds τ used for identifying sufficient input subsets in TF binding datasets. In each dataset, the threshold is defined as the 90th percentile of the predictive test distribution.

The nucleotide occurring at base position (`A`, `C`, `G`, `T`) is encoded as a one-hot representation which is fed into the CNN. Zeng et al. (2016) showed that convolutional neural network architectures outperform other models for this TF binding prediction task.

For each of the 422 prediction tasks, we employ the best-performing "1layer_128motif" architecture from Zeng et al. (2016), defined as follows:

1. **Input**: (101 x 4) sequence encoding
2. **Convolutional Layer 1**: Applies 128 kernels of window size 24, with ReLU activation
3. **Global Max Pooling Layer 1**: Performs global max pooling
4. **Dense Layer 1**: 32 neurons, with ReLU activation and dropout probability 0.5
5. **Dense Layer 2**: 1 neuron (output probability), with sigmoid activation

We hold out 1/8 of each train set for validation and minimize binary cross-entropy using the Adadelta optimizer (Zeiler, 2012) with default parameter settings in Keras (Chollet et al., 2015). We train each model on each of the 422 datasets for 10 epochs (using batch size 128) with early-stopping based on validation loss. Figure S1 shows the area under the receiver operating curve (AUC) over the 422 datasets, and we note that the performance of our models closely resembles that in Zeng et al. (2016).

S2.2 Rationale length comparison between SIS and other methods

For each dataset, we define the sufficiency threshold τ as the 90th percentile of the predictive distribution on all test sequences. The distribution of thresholds is shown in Figure S2. We compute the complete set of sufficient input subsets for each corresponding test sequence. Since A,C,G,T nucleotides all occur with similar frequency in this data, our SIS analysis simply masks each base using a uniform embedding ($[0.25, 0.25, 0.25, 0.25]$). This is also the standard strategy to represent unknown "N" nucleotides in DNA sequences that typically arise from issues in read quality. We generally find that there is only a single SIS per example for the sequences in these datasets.

On each dataset, we compute the median rationale length (as number of bases in the rationale). The distribution of median rationale length over all datasets by various methods is shown in Figure S3. Note that for the IG, LIME, and Perturb. methods, rationale length was constrained to the length of the rationales produced by our method. For the Top IG method, neither sufficiency or length constraints are enforced. We see that when the sufficiency constraint is enforced in alternative methods (Suff. IG), the rationales are significantly longer than

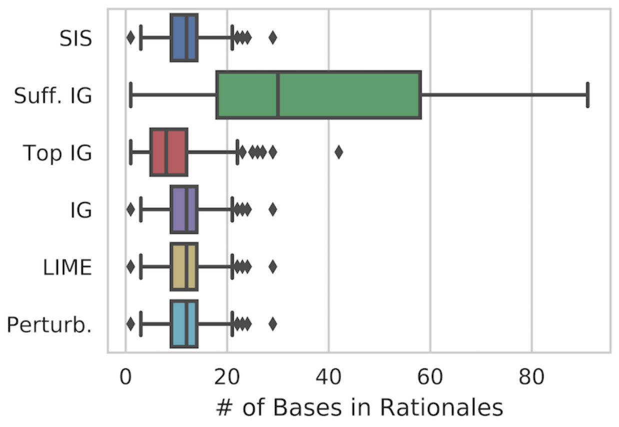


Figure S3: Length (number of bases) of rationales identified by various methods. Note that the sufficiency constraint ($f(\mathbf{x}_S) \geqslant \tau$) is only enforced for SIS and Suff. IG. The lengths of IG, LIME, and Perturb. rationales are constrained to the length of SIS rationales.

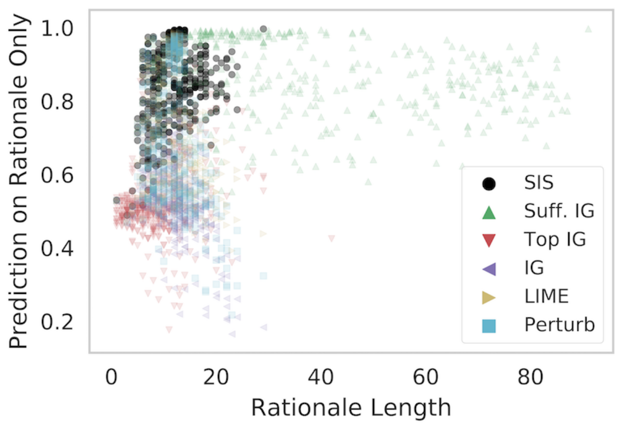


Figure S4: Prediction on rationale only (all other bases masked) vs. rationale length (number of bases) for various methods in the TF binding task.

those identified by SIS. Moreover, as shown in Figure S4, when the sufficiency constraint is not enforced (or the rationale lengths are constrained to the length of SIS rationales) in alternative methods, the rationales have significantly less predictive power, often not satisfying $f(\mathbf{x}_S) \geqslant \tau$.

S2.3 Evaluation of the quality of TF Rationales

Each rationale is padded with "N" (unknown) bases to the length of a full input sequence (101 bases) and optimally aligned with the known motif[2] according to the likelihood criterion. The aligned motif is then also padded to the same length, and we compute the divergence between between the rationale R and known motif M as:

$$\mathrm{Div}(R, M) = \sum_i D_{\mathrm{KL}}(R_i || M_i)$$

where $D_{\mathrm{KL}}(R_i||M_i) = \sum_j R_i(j) \log \frac{R_i(j)}{M_i(j)}$ is the Kullback-Leibler divergence from M_i to R_i, and M_i and R_i are distributions over bases (`A`, `C`, `G`, `T`) at position i. Note that as R and M become more dissimilar, $\mathrm{Div}(R, M)$ increases. We ensure $M_{ij} > 0 \ \forall \ i, j$ so D_{KL} is always finite.

[2]A JASPAR motif is a $n \times 4$ right stochastic matrix M. The columns represent the ACGT DNA bases and the rows a DNA sequence. It represents the marginal probability of the base j at position i being present with probability M_{ij}. The unknown base "N" receives uniform 1/4 probability for each of ACGT.

S3 Details of the MNIST Analysis

S3.1 Dataset and Model

The MNIST database of handwritten digits contains 60k training images and 10k test images (LeCun et al., 1998). All images are 28x28 grayscale, and we normalize them such that all pixel values are between 0 and 1. We use the convolutional architecture provided in the Keras MNIST CNN example.[3] The architecture is as follows:

1. **Input**: (28 x 28 x 1) image, all values $\in [0, 1]$
2. **Convolutional Layer 1**: Applies 32 3x3 filters with ReLU activation
3. **Convolutional Layer 2**: Applies 64 3x3 filters, with ReLU activation
4. **Pooling Layer 1**: Performs max pooling with a 2x2 filter and dropout probability 0.25
5. **Dense Layer 1**: 128 neurons, with ReLU activation and dropout probability 0.5
6. **Dense Layer 2**: 10 neurons (one per digit class), with softmax activation

The Adadelta optimizer (Zeiler, 2012) is used to minimize cross-entropy loss on the training set. The final model achieves 99.7% accuracy on the train set and 99.1% accuracy on the held-out test set.

S3.2 Local Minima in Backward Selection

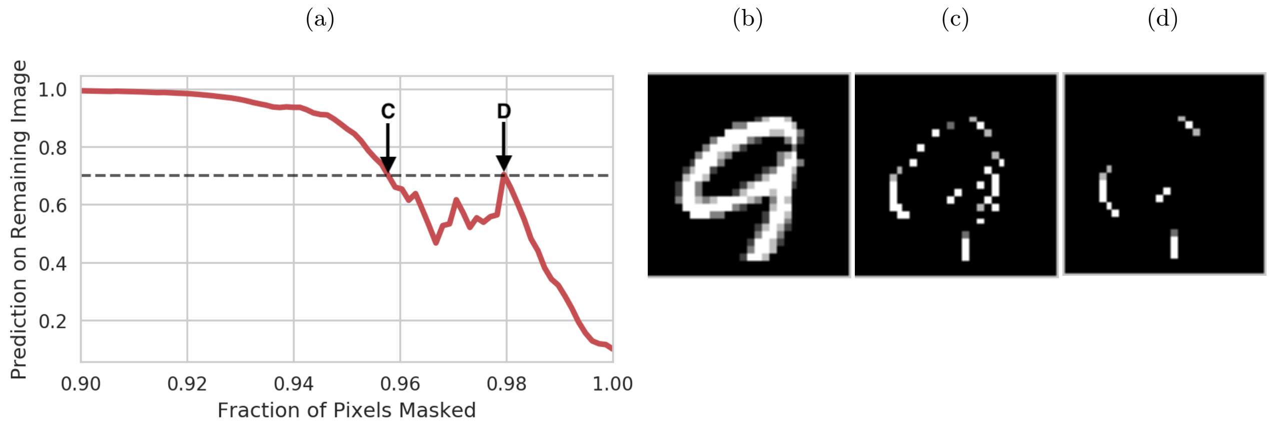


Figure S5: **(a)** Prediction on remaining image as pixels are masked during backward selection, when our CNN classifier is fed the MNIST digit in (b). The dashed line depicts the threshold $\tau = 0.7$. **(b)** Original image (class 9). **(c)** SIS if backward selection were to terminate the first time prediction on remaining image drops below 0.7, corresponding to point **C** in (a) (CNN predicts class 9 with probability 0.700 on this SIS). **(d)** Actual SIS produced by our **FindSIS** algorithm, corresponding to point **D** in (a) (CNN predicts class 9 with probability 0.704 on this SIS).

Figure S5 demonstrates an example MNIST digit for which there exists a local minimum in the backward selection phase of our algorithm to identify the initial SIS. Note that if we were to terminate the backward selection as soon as predictions drop below the decision threshold, the resulting SIS would be overly large, violating our minimality criterion. It is also evident from Figure S5 that the smaller-cardinality SIS in (d), found after the initial local optimum in (c), presents a more interpretable input pattern that enables better understanding of the core motifs influencing our classifier's decisions. To avoid suboptimal results, it is important to run a complete backward selection sweep until the entire input is masked before building the SIS upward, as done in our **SIScollection** procedure.

[3]`http://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py`

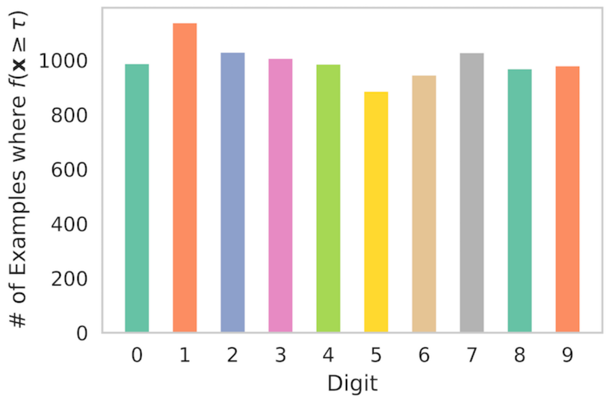


Figure S6: Number of examples per digit in the test set for which $f(\mathbf{x}) \geqslant \tau$ for the top class. The complete set of sufficient input subsets is computed for all of these examples.

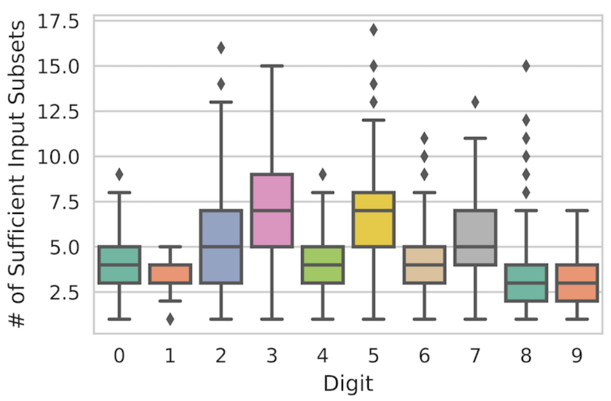


Figure S7: Distributions of number of sufficient input subsets identified per image, by digit.

S3.3 Energy Distance Between Image SIS

To cluster SIS from the image data, we compute the pairwise distance between two SIS subsets S_1 and S_2 as the energy distance (Rizzo and Székely, 2016) between two distributions over the image pixel coordinates that comprise the SIS, X_1 and $X_2 \in \mathbb{R}^2$:

$$D(X_1, X_2) = 2 \cdot \mathbb{E}\,||X_1 - X_2|| - \mathbb{E}\,||X_1 - X_1'|| - \mathbb{E}\,||X_2 - X_2'|| \geqslant 0$$

Here, X_i is uniformly distributed over the pixels that are selected as part of the SIS subset S_i, X_i' is an i.i.d. copy of X_i, and $||\cdot||$ represents the Euclidean norm. Unlike a Euclidean distance between images, our usage of the energy distance takes into account distances between the similar pixel coordinates that comprise each SIS. The energy distance offers a more efficiently computable integral probability metric than the optimal transport distance, which has been widely adopted as an appropriate measure of distance between images.

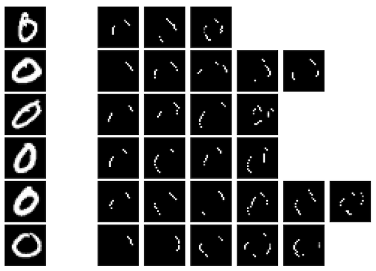
S3.4 SIS Clustering and Adversarial Analysis

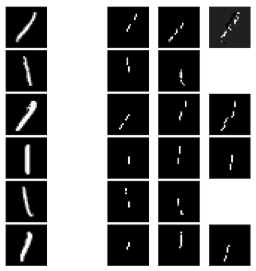
We set the threshold $\tau = 0.7$ for SIS to ensure that the model is confident in its class prediction (probability of the predicted class is $\geqslant 0.7$). Almost all test examples initially have $f(\mathbf{x}) \geqslant \tau$ for the top class (Figure S6). We identify all test examples that satisfy this condition and use SIS to identify all sufficient input subsets. The number of sufficient input subsets per digit is shown in Figure S7.

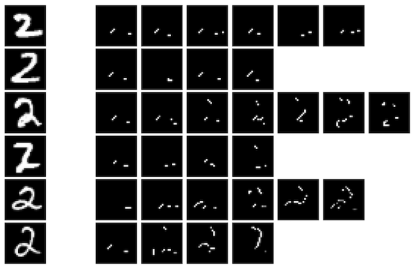
We apply our **SIScollection** algorithm to identify sufficient input subsets on MNIST test digits (Section 4.3). Examples of the complete SIS-collection corresponding to randomly chosen digits are shown in Figure S8. We also cluster all the sufficient input subsets identified for each class (Section 4.4), depicting the results in Figure S9.

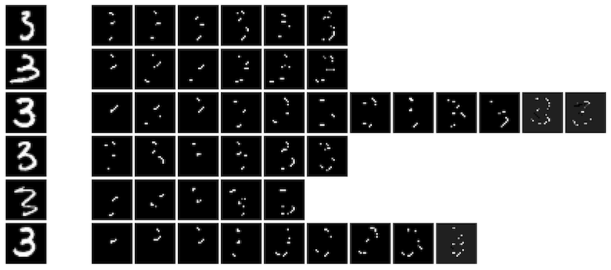
In Figure 8, we show an MNIST image of the digit 9, adversarially perturbed to 4, and the sufficient subsets corresponding to the adversarial prediction. Although a visual inspection of the perturbed image does not really reveal exactly how it has been manipulated, it becomes immediately clear from the SIS-collection for the adversarial image. These sets shows that the perturbation modifies pixels in such a way that input patterns similar to the typical SIS-collection for a 4 (Figure 7) become embedded in the image. The adversarial manipulation was done using the Carlini-Wagner L_2 (CW2) attack[4] (Carlini and Wagner, 2017b) with a confidence parameter of 10. The CW2 attack tries to find the minimal change to the image, with respect to the L_2 norm, that will lead the image to be misclassified. Carlini and Wagner (2017a) demonstrate it to be one of the strongest extant adversarial attacks.

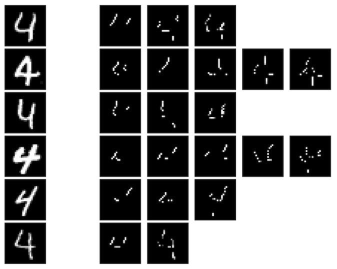
[4]Implemented in the cleverhans library of Papernot et al. (2017)

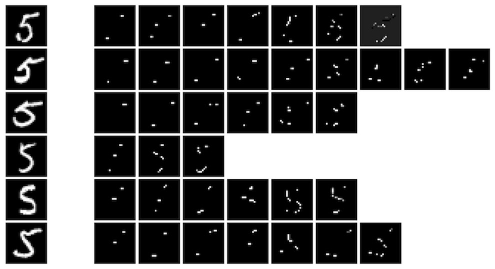

(a) Digit 0

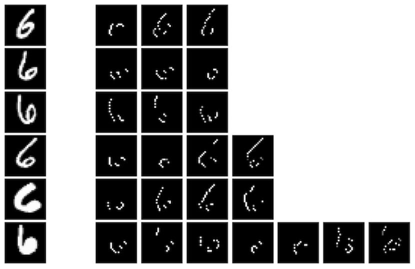

(b) Digit 1

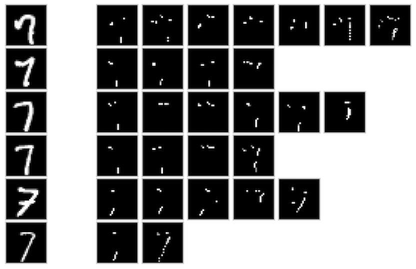

(c) Digit 2

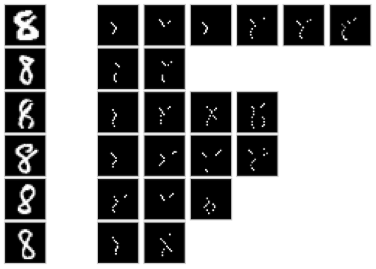

(d) Digit 3


(e) Digit 4


(f) Digit 5


(g) Digit 6


(h) Digit 7


(i) Digit 8

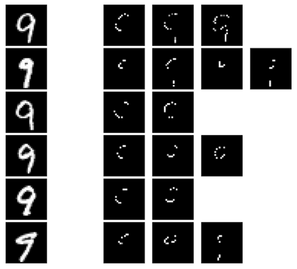
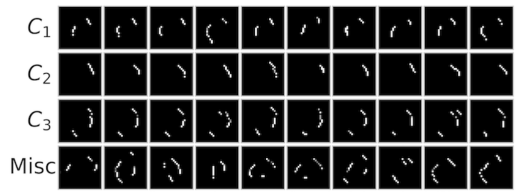
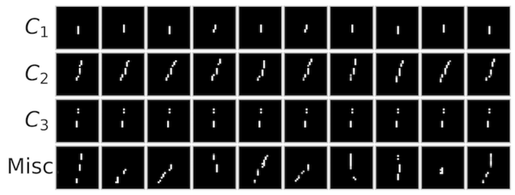

(j) Digit 9

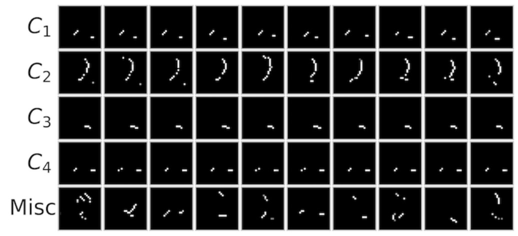
Figure S8: Visualization of SIS-collections identified from MNIST digits that are confidently classified by the CNN. For each class, six examples were chosen randomly. For each example, we show the original image (left) and the complete set of sufficient input subsets identified for that example (remaining images in each row). Each individual SIS satisfies $f(\mathbf{x}_S) \geqslant \tau$ for that class.



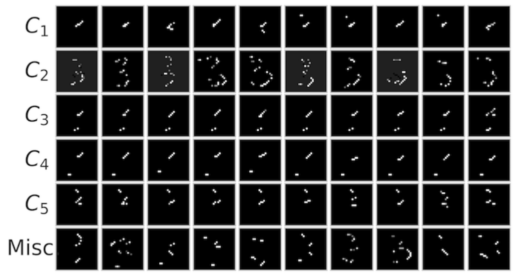
(a) Digit 0



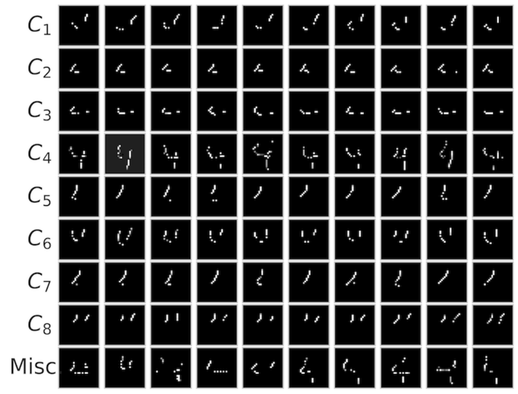
(b) Digit 1



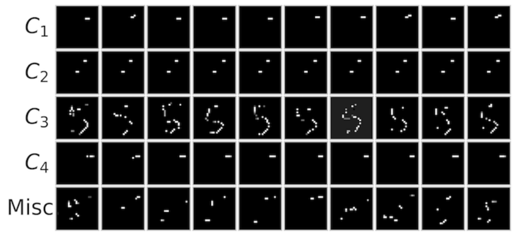
(c) Digit 2



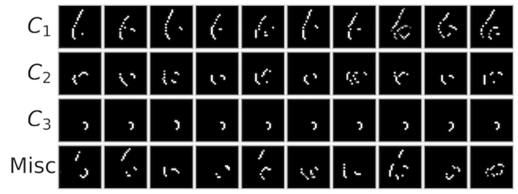
(d) Digit 3



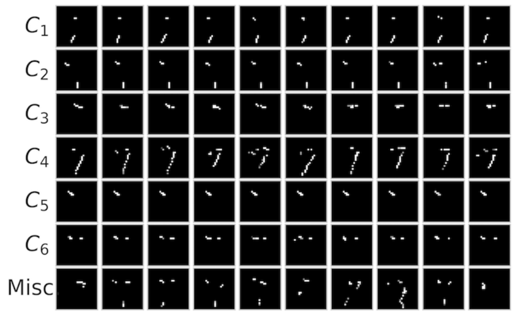
(e) Digit 4



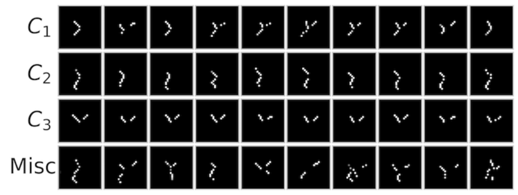
(f) Digit 5



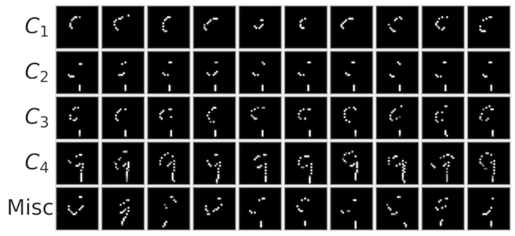
(g) Digit 6



(h) Digit 7



(i) Digit 8



(j) Digit 9

Figure S9: Clustering all the SIS found for each digit under the CNN model (see Section 4.4). Each row contains images drawn from one cluster. The bottom row ("Misc") contains a sample of miscellaneous SIS not assigned to any cluster by DBSCAN.

S3.5 Understanding Differences Between MNIST Classifiers

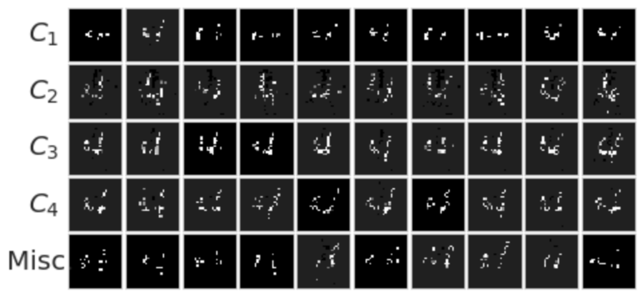
We use SIS and our clustering procedure to understand and visualize differences in features learned by two different models trained on the same MNIST digit classification task. In addition to the previously-described CNN model (see Section S3.1), we also trained a simple multilayer perceptron (MLP) on the same task. The MLP architecture is as follows:

1. **Input**: 784-dimensional (flattened) image, all values $\in [0, 1]$
2. **Dense Layer 1**: 250 neurons, ReLU activation, and dropout probability 0.2
3. **Dense Layer 2**: 250 neurons, ReLU activation, and dropout probability 0.2
4. **Dense Layer 3**: 10 neurons (one per digit class), with softmax activation

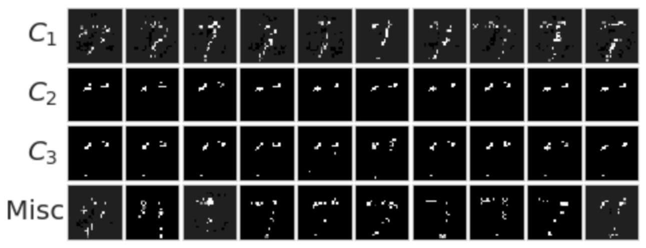
As with the CNN, Adadelta (Zeiler, 2012) is used to minimize cross-entropy loss on the training set. The final MLP model achieves 99.7% accuracy on the train set and 98.3% accuracy on the test set, which is close to the performance of the CNN (see Section S3.1).

We apply the same procedure as in Section 4.3 to extract the SIS-collection from all applicable test images using the MLP. To understand differences between the feature patterns that each model has learned to associate with predicting each digit, we combine all SIS (from both models for a particular class) and run our clustering procedure (see Section 4.4 and Figure 9). In the resulting clustering, we list what percentage of the SIS in each cluster stem from the CNN vs. the MLP. Most clusters contain examples purely from a single model, indicating the two models have learned to associate different feature patterns with the target class (Figure 9), which was chosen to be the digit 4 in this case.

For further comparison, we include clustering results for the SIS extracted from the MLP as evidence for digits 4 and 7 (Figure S10). Additionally, Figure S11 shows all of the SIS extracted from example digits from these classes applying our procedure on the MLP.

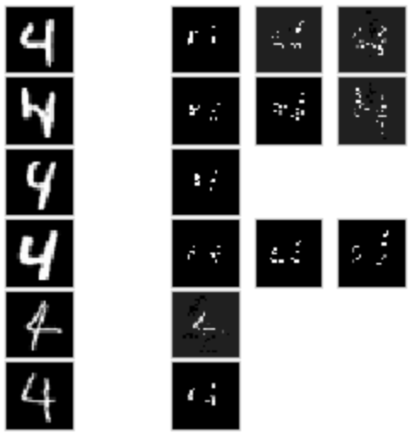


(a) Digit 4



(b) Digit 7

Figure S10: Clustering all the SIS identified by our method on digits 4 and 7 under the MLP model (see Section 4.4). Each row contains images drawn from one cluster. The bottom row ("Misc") contains a sample of miscellaneous SIS not assigned to any cluster by DBSCAN. Compare to the SIS-clustering from our CNN model (Figure S9).


(a) Digit 4

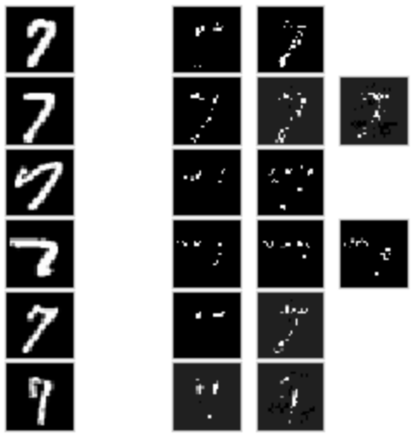

(b) Digit 7

Figure S11: Visualization of SIS-collections identified for MNIST digits 4 and 7 under the MLP model. For each class, six examples were chosen randomly. For each example, we show the original image (left) and the complete set of sufficient input subsets identified for that example (remaining images in each row). Note that each individual SIS satisfies $f(\mathbf{x}_S) \geqslant \tau$ for that class. Compare to the SIS extracted from our CNN (Figure S8).

S4 Details of the Beer Reviews Sentiment Analysis

S4.1 Beer Reviews Data Description

Following Lei et al. (2016), we use a preprocessed version of the BeerAdvocate[5] dataset[6] which contains decorrelated numerical ratings toward three aspects: *aroma*, *appearance*, and *palate* (each normalized to $[0, 1]$). Dataset statistics can be found in Table S1. Reviews were tokenized by converting to lowercase and filtering punctuation, and we used a vocabulary containing the top 10,000 most common words. McAuley et al. (2012) also provide a subset of human-annotated reviews, in which humans manually selected full sentences in each review that describe the relevant aspects. This annotated set was never seen during training and used solely as part of our evaluation.

S4.2 Model Architecture and Training

Long short-term memory (LSTM) networks are commonly employed for natural language tasks such as sentiment analysis (Wang et al., 2016; Radford et al., 2017). We use a recurrent neural network (RNN) architecture with two stacked LSTMs as follows:

1. **Input/Embeddings Layer**: Sequence with 500 timesteps, the word at each timestep is represented by a (learned) 100-dimensional embedding
2. **LSTM Layer 1**: 200-unit recurrent layer with LSTM (forward direction only)
3. **LSTM Layer 2**: 200-unit recurrent layer with LSTM (forward direction only)
4. **Dense**: 1 neuron (sentiment output), sigmoid activation

With this architecture, we use the Adam optimizer (Kingma and Ba, 2015) to minimize mean squared error (MSE) on the training set. We use a held-out set of 3,000 examples for validation (sampled at random from the pre-defined test set from Lei et al. (2016)). Our test set consists of the remaining 7,000 test examples. Training results are shown in Table S1.

Table S1: Summary and performance statistics (mean squared error (MSE) and Pearson correlation coefficient ρ) for beer reviews data and LSTM models.

Aspect	Fold	Size	MSE	Pearson ρ
Appearance	Train	80,000	0.016	0.864
	Validation	3,000	0.024	0.783
	Test	7,000	0.023	0.801
	Annotation	994	0.020	0.563
Aroma	Train	70,000	0.014	0.873
	Validation	3,000	0.024	0.767
	Test	7,000	0.025	0.756
	Annotation	994	0.021	0.598
Palate	Train	70,000	0.016	0.835
	Validation	3,000	0.029	0.680
	Test	7,000	0.028	0.694
	Annotation	994	0.016	0.592

[5] `https://www.beeradvocate.com/`

[6] `http://snap.stanford.edu/data/web-BeerAdvocate.html`

S4.3 Imputation Strategies: Mean vs. Hot-deck

In Section 3, we discuss the problem of masking input features. Here, we show that the mean-imputation approach (in which missing inputs are masked with a mean embedding, taken over the entire vocabulary) produces a nearly identical change in prediction to a nondeterministic hot-deck approach (in which missing inputs are replaced by randomly sampling feature-values from the data). Figure S12 shows the change in prediction $f(\mathbf{x} \backslash \{i\}) - f(\mathbf{x})$ by both imputation techniques after drawing a training example $\mathbf{x}$ and word $x_i \in \mathbf{x}$ (both uniformly at random) and replacing x_i with either the mean embedding or a randomly selected word (drawn from the vocabulary, based on counts in the training corpus). This procedure is repeated 10,000 times. Both resulting distributions have mean near zero ($\mu_{\text{mean-embedding}} = -7.0\text{e}{-4}$, $\mu_{\text{hot-deck}} = -7.4\text{e}{-4}$), and the distribution for mean embedding is slightly narrower ($\sigma_{\text{mean-embedding}} = 0.013$, $\sigma_{\text{hot-deck}} = 0.018$). We conclude that mean-imputation is a suitable method for masking information about particular feature values in our SIS analysis.

We also explored other options for masking word information, e.g. replacement with a zero embedding, replacement with the learned <PAD> embedding, and simply removing the word entirely from the input sequence, but each of these alternative options led to undesirably larger changes in predicted values as a result of masking, indicating they appear more informative to f than replacement via the feature-mean.

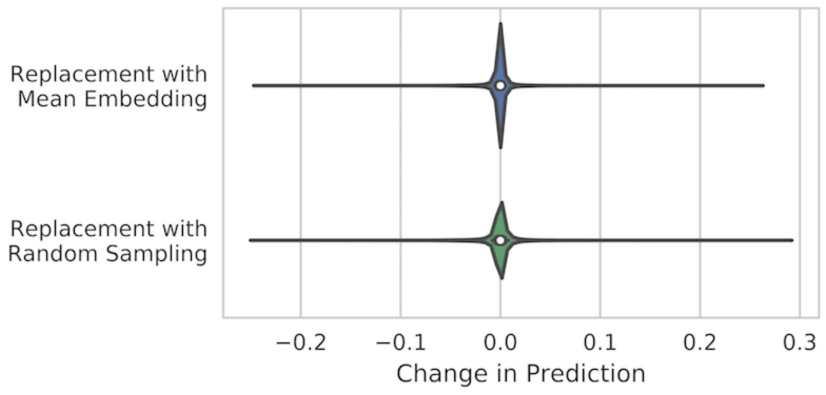


Figure S12: Change in prediction $(f(\mathbf{x} \backslash \{i\}) - f(\mathbf{x}))$ after masking a randomly chosen word with mean imputation or hot-deck imputation. 10,000 replacements were sampled from the aroma beer reviews training set.

S4.4 Feature Importance Scores

For each feature i in the input sequence, we quantify its marginal importance by individually perturbing only this feature:

$$\text{Feature Importance}(i) = \text{prediction on original input} - \text{prediction with feature } i \text{ masked} \tag{1}$$

Note that these marginal Feature Importance scores are identical to those of the Perturb. method described in Section S1. The marginal Feature Importance scores are summarized in Table S2 and Figure S13. Compared to the Suff. IG and Suff. LIME methods, our **SIScollection** technique produces rationales that are much shorter and contain fewer irrelevant (i.e. not marginally important) features (Table S2, Figures S13 and S14). Note that by construction, the rationales of the Suff. Perturb. method contain features with the greatest Feature Importance, since this precisely how the ranking in Suff. Perturb. is defined.

S4.5 Additional Results for Aroma aspect

We apply our method to the set of reviews containing sentence-level annotations. Note that these reviews (and the human annotations) were not seen during training. We choose thresholds $\tau_+ = 0.85$, $\tau_- = 0.45$ for strong positive and strong negative sentiment, respectively, and extract the complete set of sufficient input subsets using our method. Note that in our formulation above, we apply our method to inputs $\mathbf{x}$ where $f(\mathbf{x}) \geqslant \tau$. For

Table S2: Statistics for rationale length and feature importance in aroma prediction. For rationale length, median and max indicate percentage of input text in the rationale. For marginal perturbed feature importance, we indicate the median importance of features in rationales and features from the other (non-rationale) text. p-values are computed using a Wilcoxon rank-sum test.

Method	Rationale Length (% of text)			Marginal Perturbed Feature Importance		
	Med.	Max	p (vs. SIS)	Med. (Rationale)	Med. (Other)	p (vs. SIS)
SIS	**3.9%**	**17.3%**	–	0.0112	1.50e-05	–
Suff. IG	7.7%	89.7%	5e-26	0.0068	1.85e-05	3e-42
Suff. LIME	7.2%	84.0%	4e-23	0.0075	1.87e-05	1e-35
Suff. Perturb.	5.1%	18.3%	1e-06	0.0209	1.90e-05	1e-72

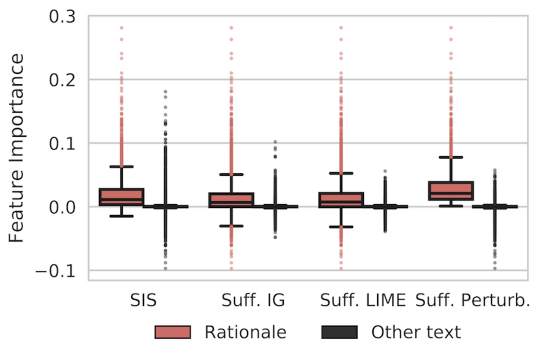


Figure S13: Importance of individual features in the rationales for aroma prediction in beer reviews

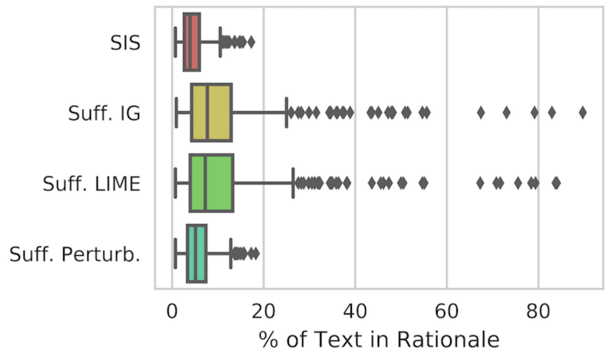


Figure S14: Length of rationales for aroma prediction

the sentiment analysis task, we analogously apply our method for both $f(\mathbf{x}) \geqslant \tau_+$ and $-f(\mathbf{x}) \geqslant -\tau_-$, where the model predicts either strong positive or strong negative sentiment, respectively. These thresholds were set empirically such that they were sufficiently apart, based on the distribution of predictions (Figure S15). For most reviews, **SIScollection** outputs just one or two SIS sets (Figure S16).

We analyzed the predictor output following the elimination of each feature in the **BackSelect** procedure (Section 3). Figure S17 shows the LSTM output on the remaining unmasked text $f(\mathbf{x}_{S\setminus\{i*\}})$ at each iteration of **BackSelect**, for all examples. This figure reveals that only a small number of features are needed by the model in order to make a strong prediction (most features can be removed without changing the prediction). We see that as those final, critical features are removed, there is a rapid, monotonic decrease in output values. Finally, we see that the first features to be removed by **BackSelect** are those which generally provide negative evidence against the decision.

S4.6 Understanding Differences Between Sentiment Predictors

We demonstrate how our SIS-clustering procedure can be used to understand differences in the types of concepts considered important by different neural network architectures. In addition to the LSTM (see Section S4.2), we trained a convolutional neural network (CNN) on the same sentiment analysis task (on the aroma aspect). The CNN architecture is as follows:

1. **Input/Embeddings Layer**: Sequence with 500 timesteps, the word at each timestep is represented by a (learned) 100-dimensional embedding
2. **Convolutional Layer 1**: Applies 128 filters of window size 3 over the sequence, with ReLU activation
3. **Max Pooling Layer 1**: Max-over-time pooling, followed by flattening, to produce a (128,) representation
4. **Dense**: 1 neuron (sentiment output), sigmoid activation

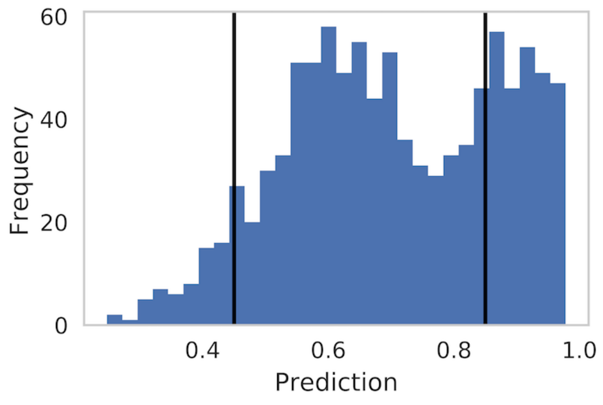


Figure S15: Predictive distribution on the annotation set (held-out) using the LSTM model for aroma. Vertical lines indicate decision thresholds ($\tau_+ = 0.85$, $\tau_- = 0.45$) selected for **SIScollection**.

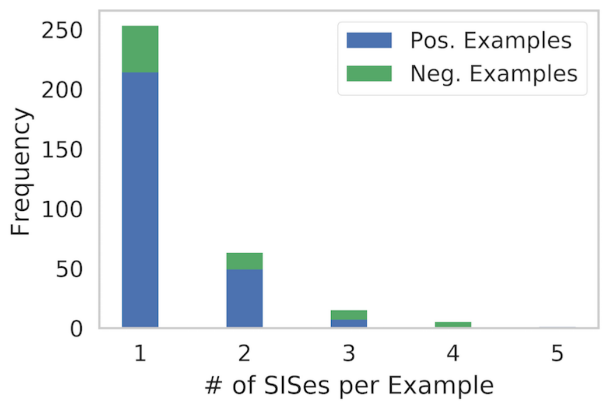


Figure S16: Number of sufficient input subsets for aroma identified by **SIScollection** per example.

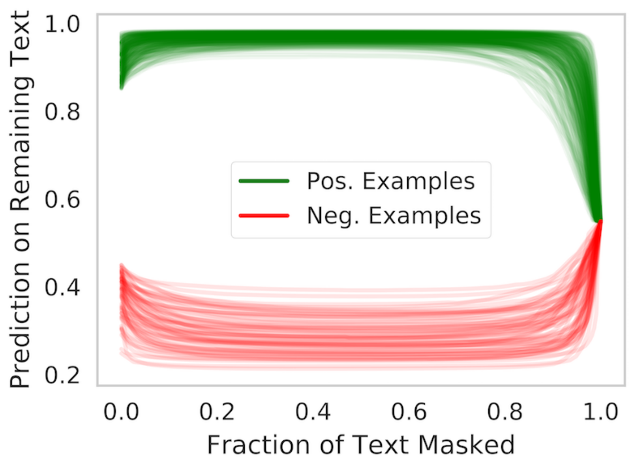
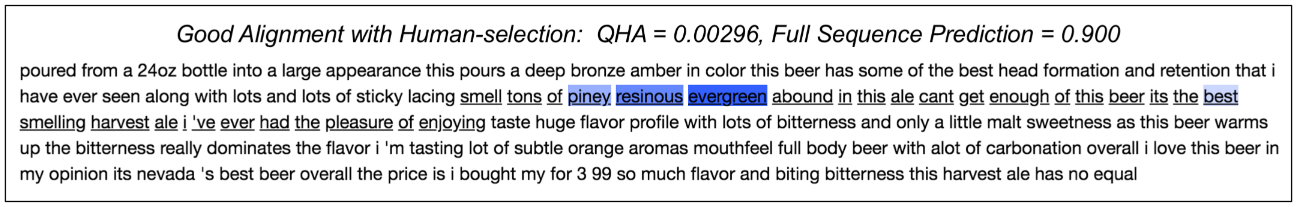
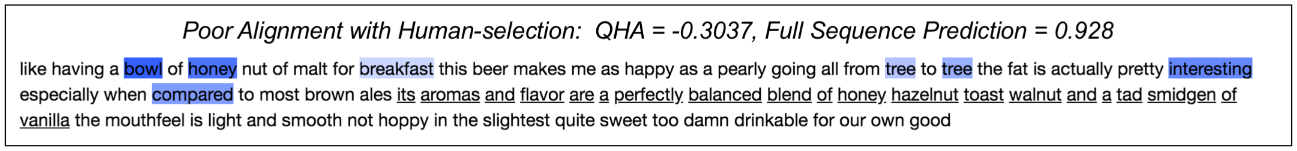


Figure S17: Prediction history on remaining (unmasked) text at each step of the **BackSelect** procedure, for examples where aroma sentiment is predicted.



Good Alignment with Human-selection: QHA = 0.00296, Full Sequence Prediction = 0.900

poured from a 24oz bottle into a large appearance this pours a deep bronze amber in color this beer has some of the best head formation and retention that i have ever seen along with lots and lots of sticky lacing smell tons of piney resinous evergreen abound in this ale cant get enough of this beer its the best smelling harvest ale i 've ever had the pleasure of enjoying taste huge flavor profile with lots of bitterness and only a little malt sweetness as this beer warms up the bitterness really dominates the flavor i 'm tasting lot of subtle orange aromas mouthfeel full body beer with alot of carbonation overall i love this beer in my opinion its nevada 's best beer overall the price is i bought my for 3 99 so much flavor and biting bitterness this harvest ale has no equal



Poor Alignment with Human-selection: QHA = -0.3037, Full Sequence Prediction = 0.928

like having a bowl of honey nut of malt for breakfast this beer makes me as happy as a pearly going all from tree to tree the fat is actually pretty interesting especially when compared to most brown ales its aromas and flavor are a perfectly balanced blend of honey hazelnut toast walnut and a tad smidgen of vanilla the mouthfeel is light and smooth not hoppy in the slightest quite sweet too damn drinkable for our own good

Figure S18: Beer reviews (aroma) in which human-selected sentences (underlined) are aligned well (top) and poorly (bottom) with predictive model. Fraction of SIS in the human sentences corresponds accordingly. In the bottom example (poor alignment between human-selection and predictive model), our procedure has surfaced a case where the LSTM has learned features that diverge from what a human would expect (and may suggest overfitting).

Table S3: All clusters of sufficient input subsets extracted from reviews from the test set predicted to have positive aroma by the LSTM. Frequency indicates the number of occurrences of the SIS in the cluster.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1	smell amazing wonderful	2	nice wonderful nose	2	wonderful amazing	2	amazing amazing	2
C_2	grapefruit mango pineapple	2	pineapple grapefruit pineapple grapefruit	1	hops grapefruit pineapple floyds	1	mango pineapple incredible	1
C_3	nice smell citrus nice grapefruit taste	1	smell great complex ripe taste	1	nice smell nice hop smell pine taste	1	love nice nice smell bliss taste	1
C_4	fresh great fantastic taste	1	rich great fantastic hoped	1	fantastic cherries fantastic	1	everyone great snifters fantastic	1
C_5	awesome bounds	1	awesome grapefruit awesome	1	awesome awesome pleasing	1	awesome nailed nailed	1
C_6	creme brulee brulee	3	creme brulee decadent	1	incredible creme brulee	1	creme brulee exceptional	1
C_7	oak vanilla chocolate cinnamon vanilla oak love	1	dose oak chocolate vanilla acidic	1	vanilla figs oak thinner great	1	chocolate aroma oak vanilla dessert	1

Table S4: All clusters of sufficient input subsets extracted from reviews from the test set predicted to have negative aroma by the LSTM. Frequency indicates the number of occurrences of the SIS in the cluster. Dashes are used in clusters with under 4 unique SIS.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1	awful	15	skunky skunky	9	skunky t	7	skunky taste	6
C_2	garbage	3	taste garbage	1	garbage avoid	1	garbage rice	1
C_3	vomit	16	-	-	-	-	-	-
C_4	gross rotten	1	rotten forte	1	awkward rotten	1	rotten offputting	1
C_5	rancid horrid	1	rancid t	1	rancid	1	rancid avoid	1
C_6	rice t rice	2	rice rice	1	rice tasteless	1	budweiser rice	1

Note that a new set of embeddings was learned with the CNN. As with the LSTM model, we use Adam (Kingma and Ba, 2015) to minimize MSE on the training set. For the aroma aspect, this CNN achieves 0.016 (0.850), 0.025 (0.748), 0.026 (0.741), 0.014 (0.662) MSE (and Pearson ρ) on the Train, Validation, Test, and Annotation sets, respectively. We note that this performance is very similar to that from the LSTM (see Table S1).

We apply our procedure to extract the SIS-collection from all applicable test examples using the CNN, as in Section 4.1. Figure 10a shows the predictions from one model (LSTM or CNN) when fed input examples that are SIS extracted with respect to the *other* model (for reviews predicted to have positive sentiment toward the aroma aspect). For example, in Figure 10a, "CNN SIS Preds by LSTM" refers to predictions made by the LSTM on the set of sufficient input subsets produced by applying our **SIScollection** procedure on all examples $\mathbf{x} \in \mathcal{X}_{\text{test}}$ for which $f_{\text{CNN}}(\mathbf{x}) \geqslant \tau_+$.[7] Since the word embeddings are model-specific, we embed each SIS using the embeddings of the model making the prediction (note that while the embeddings are different, the vocabulary is the same across the models).

In Table 2, we show five example clusters (and cluster composition) resulting from clustering the combined set of all sufficient input subsets extracted by the LSTM and CNN on reviews in the test set for which a model predicts positive sentiment toward the aroma aspect. The complete clustering on reviews receiving positive sentiment predictions is shown in Table S5 and in Table S6 for reviews receiving negative sentiment predictions.

Table S5: Joint clustering of the SIS extracted from beer reviews predicted to have positive aroma by LSTM or CNN model. Frequency indicates the number of occurrences of the SIS in the cluster. Percentages quantify SIS per cluster from the LSTM. Dashes are used in clusters with under 4 unique SIS.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1 (LSTM: 20%)	rich chocolate	13	very rich	9	chocolate complex	5	smells rich	4
C_2 (LSTM: 21%)	great	248	amazing	119	wonderful	112	fantastic	75
C_3 (LSTM: 47%)	best smelling	23	pineapple mango	6	mango pineapple	6	pineapple grapefruit	5
C_4 (LSTM: 5%)	excellent	42	excellent flemish flemish	1	excellent excellent phenomenal	1	-	-
C_5 (LSTM: 33%)	oak chocolate	2	chocolate raisins raisins oak bourbon	1	chocolate oak	1	raisins chocolate	1
C_6 (LSTM: 5%)	goodness	19	watering goodness	1	-	-	-	-
C_7 (LSTM: 24%)	pumpkin pie	25	huge pumpkin aroma pumpkin pie	1	aroma perfect pumpkin pie taste	1	smell pumpkin nutmeg cinnamon pie	1
C_8 (LSTM: 5%)	jd	13	tremendous	8	tremendous jd	1	-	-
C_9 (LSTM: 40%)	brulee	14	creme brulee brulee	3	creme creme	1	creme brulee amazing	1
C_{10} (LSTM: 0%)	s wow	20	-	-	-	-	-	-
C_{11} (LSTM: 0%)	delicious	56	-	-	-	-	-	-
C_{12} (LSTM: 0%)	very nice	23	-	-	-	-	-	-
C_{13} (LSTM: 70%)	complex aroma	5	aroma complex peaches complex	1	aroma complex interesting cherries	1	aroma complex	1

[7] For experiments involving clustering and/or comparing different models, we use examples drawn from the Test fold (instead of Annotation fold, see Table S1) to consider a larger number of examples.

Table S6: Joint clustering of the SIS extracted from beer reviews predicted to have negative aroma by LSTM or CNN model. Frequency indicates the number of occurrences of the SIS in the cluster. Percentages quantify SIS per cluster from the LSTM. Dashes are used in clusters with under 4 unique SIS.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1 (LSTM: 29%)	not	247	no	105	bad	104	macro	94
C_2 (LSTM: 100%)	gross rotten	1	-	-	-	-	-	-
C_3 (LSTM: 100%)	rotten garbage	1	-	-	-	-	-	-
C_4 (LSTM: 62%)	vomit	26	-	-	-	-	-	-
C_5 (LSTM: 21%)	budweiser	22	sewage budweiser	1	metal budweiser	1	budweiser budweiser budweiser	1
C_6 (LSTM: 100%)	garbage rice	1	-	-	-	-	-	-
C_7 (LSTM: 3%)	n't	19	adjuncts	14	n't adjuncts	1	-	-
C_8 (LSTM: 0%)	faint	82	-	-	-	-	-	-
C_9 (LSTM: 0%)	adjunct	42	-	-	-	-	-	-

S4.7 Results for Appearance and Palate aspects

For posterity, we include results here from repeating the analysis in our paper for the two other non-aroma aspects measured in the beer reviews data: appearance and palate.

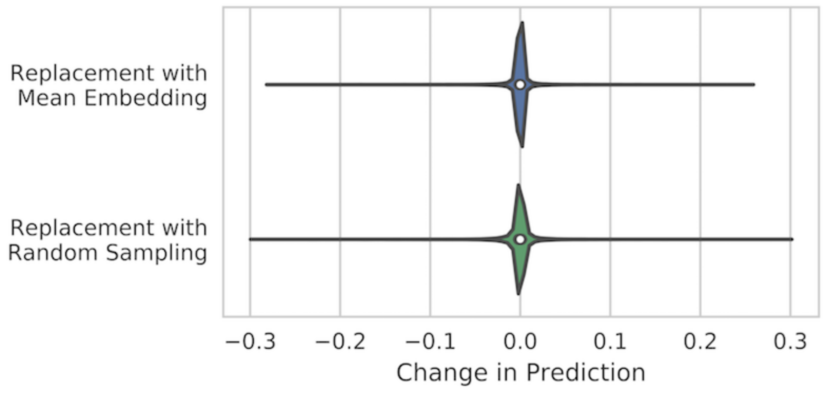


Figure S19: Change in appearance prediction $(f(\mathbf{x}\backslash\{i\}) - f(\mathbf{x}))$ after masking a randomly chosen word with mean imputation or hot-deck imputation. 10,000 replacements were sampled from the appearance beer reviews training set.

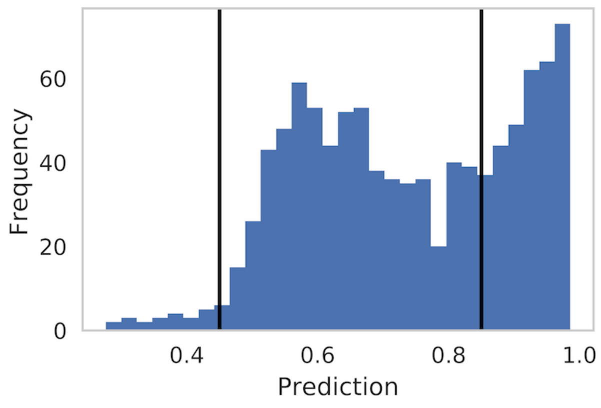


Figure S20: Predictive distribution on the annotation set (held-out) using the LSTM model for appearance. Vertical lines indicate decision thresholds ($\tau_+ = 0.85$, $\tau_- = 0.45$) selected for **SIScollection**.

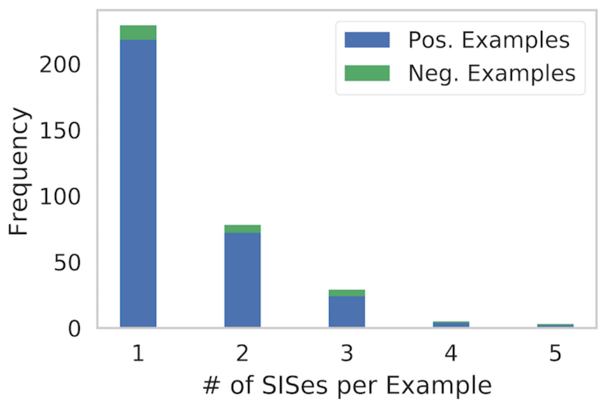


Figure S21: Number of sufficient input subsets for appearance identified by **SIScollection** per example.

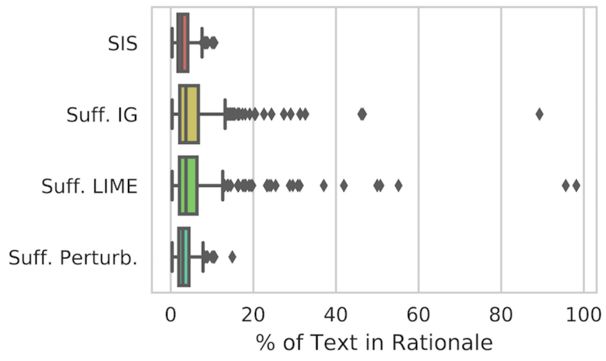


Figure S22: Length of rationales for appearance prediction

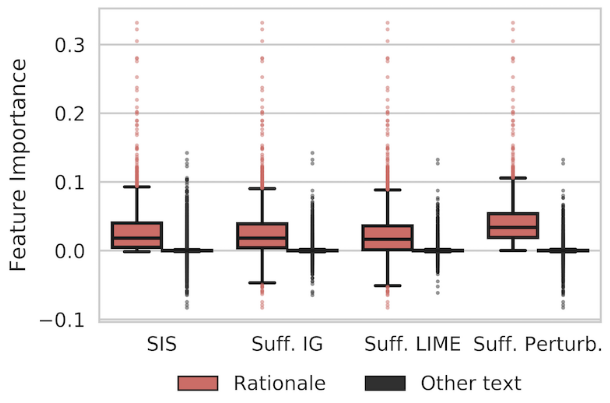


Figure S23: Importance of individual features for appearance prediction in beer review

Table S7: Statistics for rationale length and feature importance in appearance prediction. For rationale length, median and max indicate percentage of input text in the rationale. For marginal perturbed feature importance, we indicate the median importance of features in rationales and features from the other (non-rationale) text. p-values are computed using a Wilcoxon rank-sum test.

Method	Rationale Length (% of text)			Marginal Perturbed Feature Importance		
	Med.	Max	p (vs. SIS)	Med. (Rationale)	Med. (Other)	p (vs. SIS)
SIS	**2.6%**	**10.6%**	–	0.0183	1.72e-05	–
Suff. IG	3.7%	89.3%	2e-09	0.0184	2.41e-05	1e-02
Suff. LIME	3.7%	98.2%	8e-09	0.0167	2.38e-05	6e-09
Suff. Perturb.	3.0%	14.9%	9e-03	0.0339	2.51e-05	5e-44

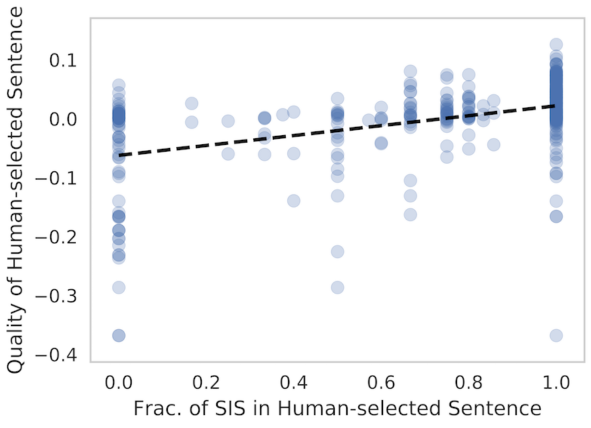


Figure S24: QHS vs. fraction of SIS in human rationale for appearance prediction

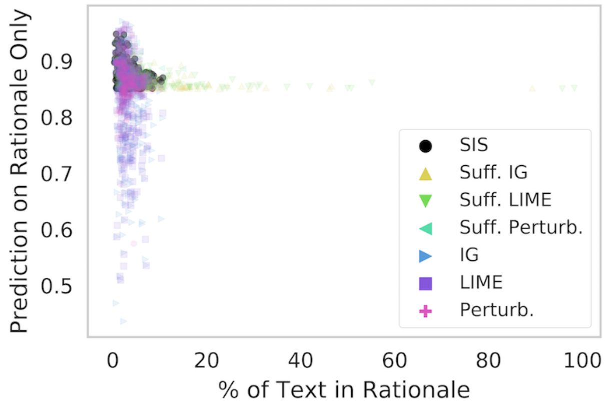


Figure S25: Prediction on rationales only vs. rationale length for various methods in positive sentiment examples for appearance. The threshold for sufficiency was $\tau_+ = 0.85$.

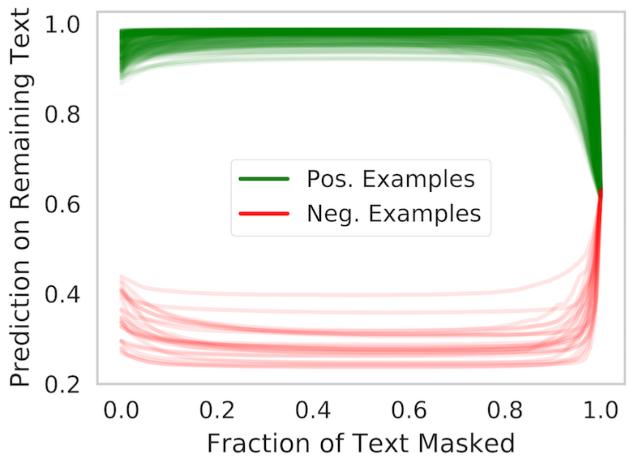


Figure S26: Prediction history on remaining (unmasked) text at each step of the **BackSelect** procedure, for examples where appearance sentiment is predicted.

Table S8: All clusters of sufficient input subsets extracted from reviews from the test set predicted to have positive appearance by the LSTM. Frequency indicates the number of occurrences of the SIS in the cluster. Dashes are used in clusters with under 4 unique SIS.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1	beautiful	376	nitro	51	looks great	38	great looking	32
C_2	gorgeous	83	-	-	-	-	-	-
C_3	beautifully	7	absolutely beautifully	2	beautifully pillowy	1	beautifully bands	1
C_4	brilliant	5	brilliant slowly	1	wonderfully brilliant	1	appearance brilliant	1
C_5	lovely looking	3	black lovely	3	impressive lovely	1	lovely crystal	1

Table S9: All clusters of sufficient input subsets extracted from reviews from the test set predicted to have negative appearance by the LSTM. Frequency indicates the number of occurrences of the SIS in the cluster. Dashes are used in clusters with under 4 unique SIS.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1	piss	46	zero	38	water water	37	water	27
C_2	unappealing	12	floaties	12	floaties unappealing	1	-	-
C_3	ugly	12	-	-	-	-	-	-

Table S10: Statistics for rationale length and feature importance in palate prediction. For rationale length, median and max indicate percentage of input text in the rationale. For marginal perturbed feature importance, we indicate the median importance of features in rationales and features from the other (non-rationale) text. p-values are computed using a Wilcoxon rank-sum test.

Method	Rationale Length (% of text)			Marginal Perturbed Feature Importance		
	Med.	Max	p (vs. SIS)	Med. (Rationale)	Med. (Other)	p (vs. SIS)
SIS	**2.4%**	13.7%	–	0.0210	-8.94e-07	–
Suff. IG	3.2%	56.1%	2e-06	0.0163	-9.54e-07	6e-10
Suff. LIME	3.0%	57.0%	7e-06	0.0173	-1.19e-06	2e-07
Suff. Perturb.	2.8%	**11.8%**	3e-03	0.0319	-1.25e-06	5e-26

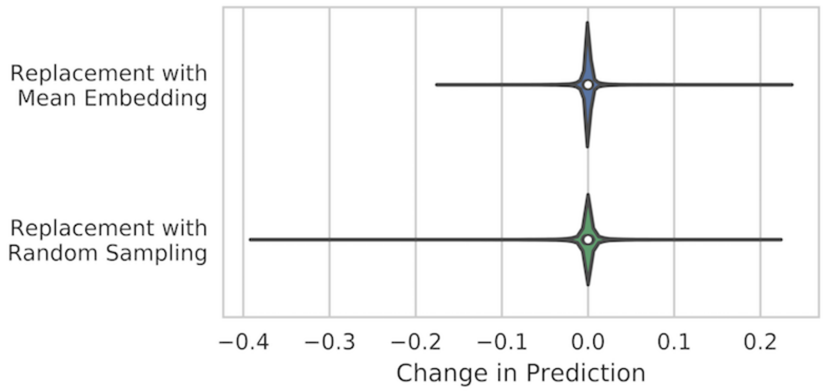


Figure S27: Change in palate prediction $(f(\mathbf{x}\backslash\{i\}) - f(\mathbf{x}))$ after masking a randomly chosen word with mean imputation or hot-deck imputation. 10,000 replacements were sampled from the palate beer reviews training set.

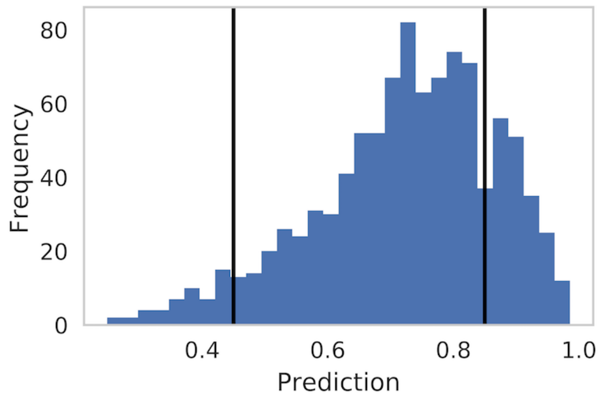


Figure S28: Predictive distribution on the annotation set (held-out) using the LSTM model for palate. Vertical lines indicate decision thresholds ($\tau_+ = 0.85$, $\tau_- = 0.45$) selected for **SIScollection**.

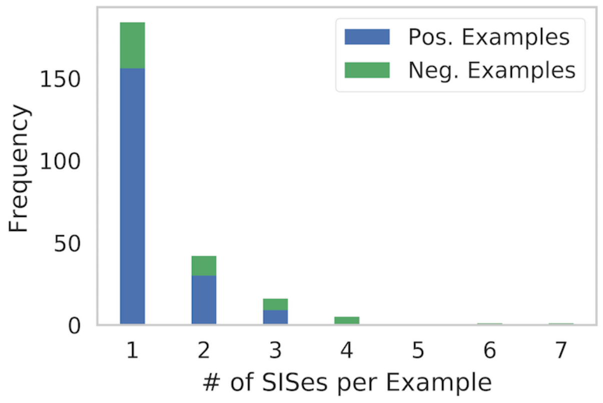


Figure S29: Number of sufficient input subsets for palate identified by **SIScollection** per example.

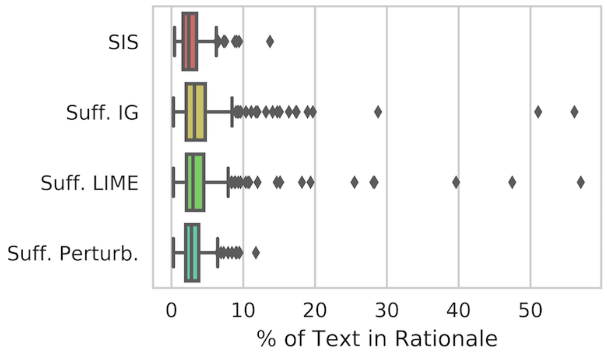


Figure S30: Length of rationales for palate prediction

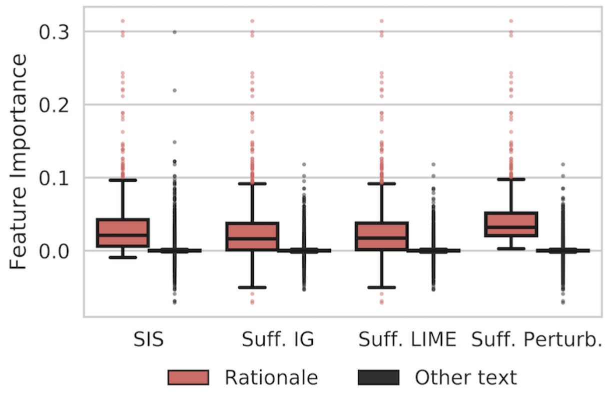


Figure S31: Importance of individual features in beer review palate rationales

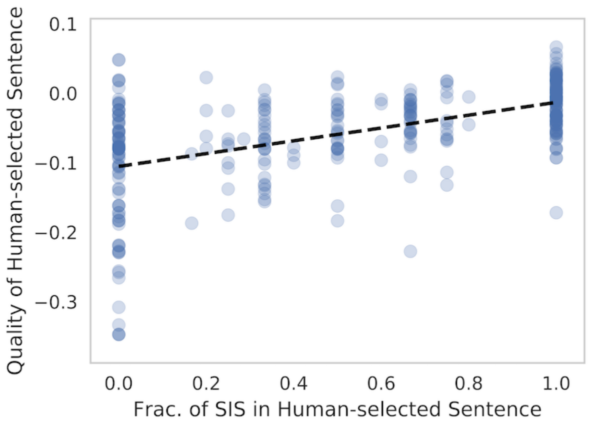


Figure S32: QHS vs. fraction of SIS in human rationale for palate prediction

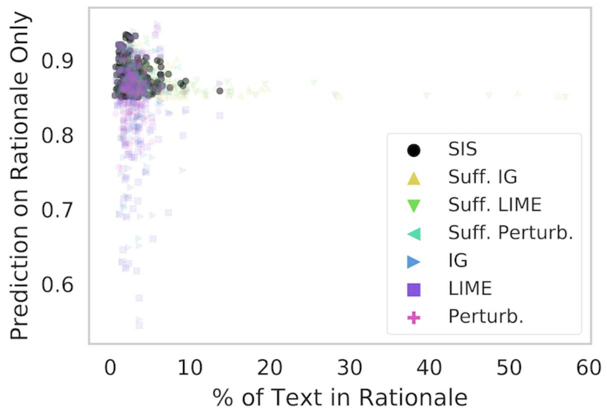


Figure S33: Prediction on rationales only vs. rationale length for various methods in positive sentiment examples for palate. The threshold for sufficiency was $\tau_+ = 0.85$.

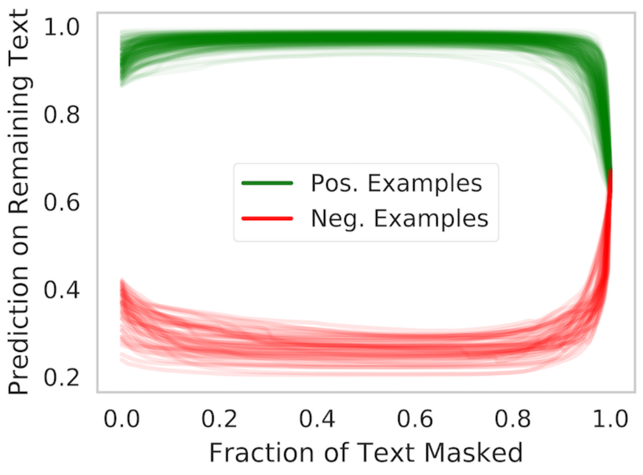


Figure S34: Prediction history on remaining (unmasked) text at each step of the **BackSelect** procedure, for examples where palate sentiment is predicted.

Table S11: All clusters of sufficient input subsets extracted from reviews from the test set predicted to have positive palate by the LSTM. Frequency indicates the number of occurrences of the SIS in the cluster. Dashes are used in clusters with under 4 unique SIS.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1	smooth creamy	27	silky smooth	20	mouthfeel perfect	16	creamy perfect	12
C_2	mouthfeel exceptional	6	exceptional mouthfeel	4	-	-	-	-
C_3	perfect	50	perfect perfect	6	-	-	-	-
C_4	smooth velvety	6	velvety smooth	6	-	-	-	-
C_5	silk	11	-	-	-	-	-	-
C_6	smooth perfect	8	mouth smooth perfect	1	perfect smooth	1	-	-
C_7	perfect great	5	great perfect	2	feels perfect	2	perfect feels great	1

Table S12: All clusters of sufficient input subsets extracted from reviews from the test set predicted to have negative palate by the LSTM. Frequency indicates the number of occurrences of the SIS in the cluster.

Cluster	SIS #1	Freq.	SIS #2	Freq.	SIS #3	Freq.	SIS #4	Freq.
C_1	overcarbonated	12	mouthfeel overcarbonated	3	way overcarbonated	1	overcarbonated mouthfeel	1
C_2	watery	302	thin	238	flat	118	mouthfeel thin	33
C_3	too carbonation masks	1	too carbonation d	1	mouthfeel odd too too	1	too carbonated admire	1
C_4	lack carbonation	4	carbonation lack	4	carbonation hurts	2	issue lack hurts	1

Additional References for the Supplementary Information

Carlini, N. and Wagner, D. (2017a). Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security.*

Carlini, N. and Wagner, D. (2017b). Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy.*

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Consortium, E. P. et al. (2012). An integrated encyclopedia of dna elements in the human genome. *Nature*, 489(7414):57.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations.*

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lei, T., Barzilay, R., and Jaakkola, T. (2016). Rationalizing neural predictions. In *Empirical Methods in Natural Language Processing.*

McAuley, J., Leskovec, J., and Jurafsky, D. (2012). Learning attitudes and attributes from multi-aspect reviews. In *IEEE International Conference on Data Mining*, pages 1020–1025.

Papernot, N., Carlini, N., Goodfellow, I., Feinman, R., Faghri, F., Matyasko, A., Hambardzumyan, K., Juang, Y.-L., Kurakin, A., Sheatsley, R., Garg, A., and Lin, Y.-C. (2017). cleverhans v2.0.0: an adversarial machine learning library. *arXiv:1610.00768.*

Radford, A., Jozefowicz, R., and Sutskever, I. (2017). Learning to generate reviews and discovering sentiment. *arXiv:1704.01444.*

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Rizzo, M. L. and Székely, G. J. (2016). Energy distance. *Wiley Interdisciplinary Reviews: Computational Statistics*, 8(1):27–38.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *International Conference on Machine Learning.*

Wang, Y., Huang, M., Zhao, L., et al. (2016). Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.*

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *arXiv:1212.5701.*

Zeng, H., Edwards, M. D., Liu, G., and Gifford, D. K. (2016). Convolutional neural network architectures for predicting dna–protein binding. *Bioinformatics*, 32(12):i121.