Web Appendix

W1 Missing Data Concern

We have access to data from only one online retailer instead of the user-centric data across all e-commerce sites. It is possible that consumers gather information and undertake transactions at many other sources during their purchase journey. We believe several facts could mitigate the concern that the competition among different retailers is missing in our analysis. First, this online retailer is a leading e-commerce site in the UK with a very high market share.[1] It is well known for its large-volume consumer reviews. The users in our sample all have loyalty membership accounts with the company, so it is less likely for these loyal customers to conduct comparison-shopping across sites. Second, the ultimate goal of this paper is to identify the impact of review content on sales. We use the regression discontinuity in time (RDIT) identification strategy. This identification strategy implies that even if consumers shop or read reviews on multiple e-commerce sites, as long as the consumers' behaviors outside of the focal site are not systematically different before and after a new review is posted to the focal website, the estimated impact of review content on conversion will be unbiased. Because we cannot think of any strong argument for the systematic difference, we believe that our identification strategy is immune to the competitive effects. Last, although Section "Descriptions of Consumers' Review-Reading Behaviors" may provide an incomplete profile of consumers' entire online decision journeys, this section provides only descriptive analysis of the data. The missing data problem will not affect our estimate of the impact of review content on conversion, because we eliminate from the regressions the journeys in which consumers do not read reviews.

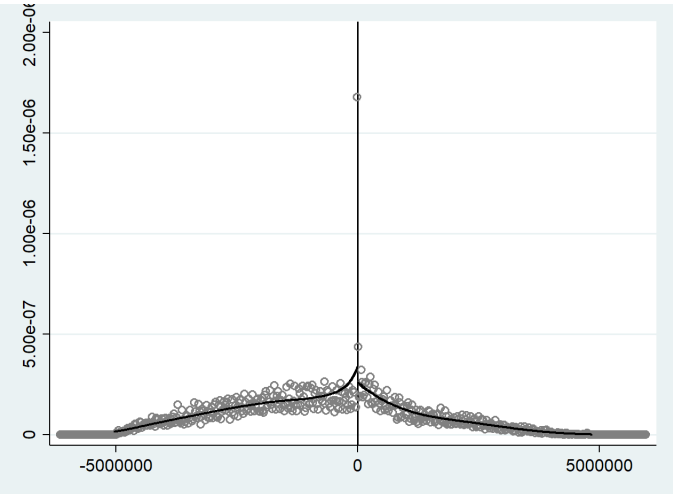
[1] Due to the non-disclosure agreement, we cannot reveal the identity of the company or its major financial statistics.

W2 Robustness Checks for the Regression Discontinuity in Time Design

Because we have multiple treatment variables for all the content dimensions, below we provide only one example for the positive aesthetics content dimension. Robustness checks for other content dimensions are available from the authors upon request.

We first assess the possibility of manipulation of the assignment variable, new review post time, by showing its distribution in Figure W1. The underlying assumption that generates the local random assignment result is that each consumer has imprecise control over the assignment variable. We can test this by checking whether the aggregate distribution of the assignment variable is discontinuous. Figure W1 shows no evidence of discontinuity at the cutoff point 0.

Figure W1: Density of the Assignment Variable: Time



Moreover, we plot a parallel RD estimated on control variables to demonstrate continuity. In Figure W2, we create the regression discontinuity plot for one of the covariates, price. In contrast to the discontinuous jump for conversion rate in Figure 5, price is continuously distributed before and after the cutoff point. This confirms the validity of the assumption that there is no precise manipulation or sorting of the assignment variable.

Figure W2: Parallel Regression Discontinuity: Price by Time

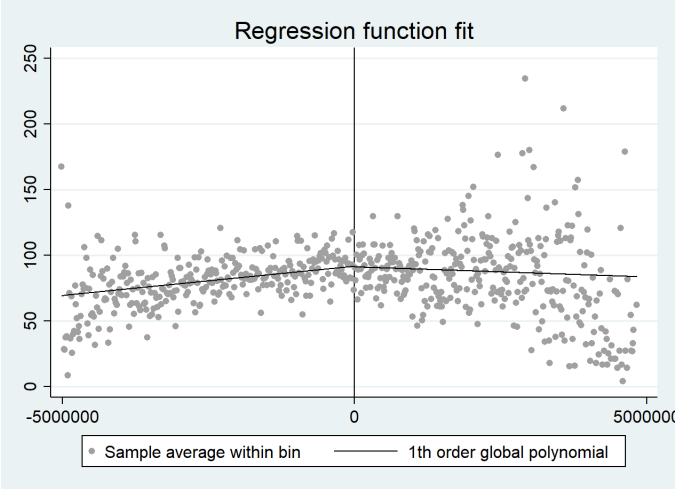
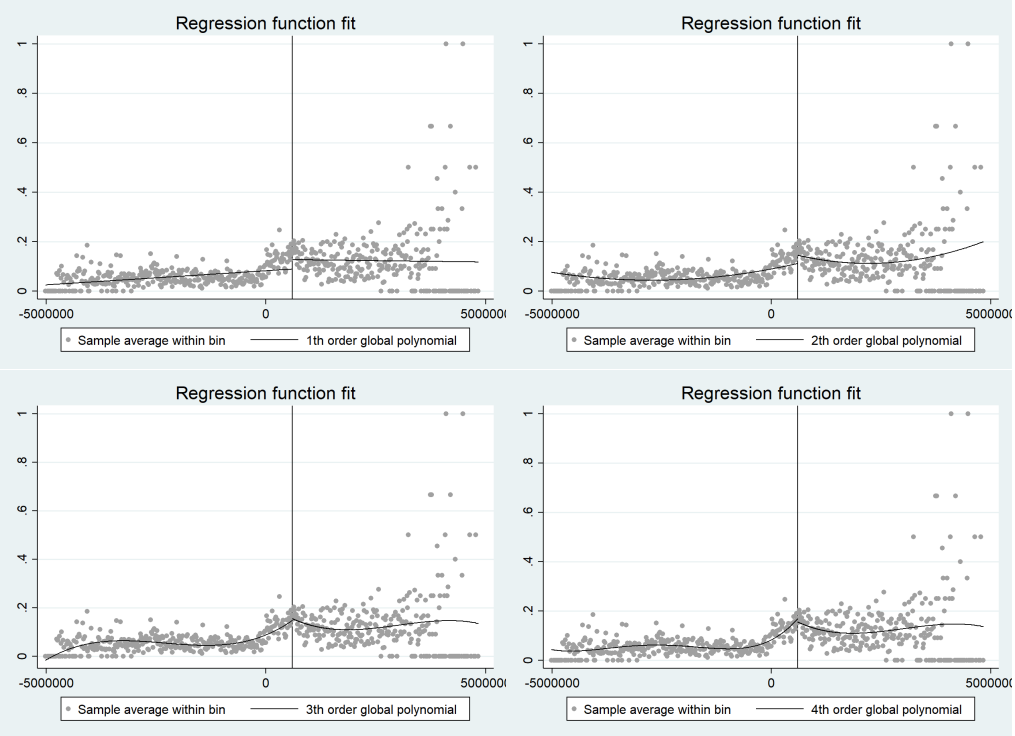


Figure W3: Placebo Test



We also conduct a placebo test by estimating a parallel RD on a different date other than the new review (with positive aesthetics information) post date. The idea of the placebo test is that if RDiT does not work, we should observe non-zero and statistically significant jumps at other discretionary cutoff points. We would then conclude that there is something wrong with RDiT. In Figure W3,

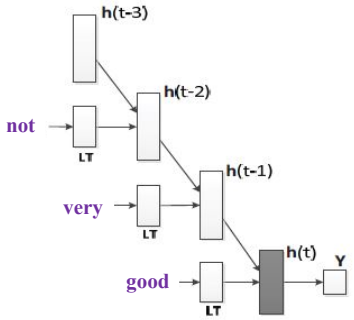
we present the regression discontinuity plots with polynomial orders 1 to 4 when the cutoff point is set at 600,000 seconds after the cutoff time stamp. The plots suggest that there is no discontinuity in conversion rate at a different cutoff point other than zero, which reassures the validity of our RDiT design.

W3 Three Deep Learning Algorithms

For each example, consider a phrase that may appear in a review text: "but not very good."

Recurrent Neural Networks – Long Short-Term Memory (LSTM)

Figure W4: Recurrent Neural Networks – Long Short-Term Memory.



From "Predicting polarities of tweets by composing word embeddings with long short-term memory," by Wang et al. 2015, Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Vol. 1, pp. 1343-1353). Copyright 2015 by the Proceedings.com. Adapted with permission.

The first deep learning algorithm we implement is long short-term memory recurrent neural networks (Hochreiter and Schmidhuber, 1997), which works by taking word or character *sequences as inputs,* and can simulate interactions of words in the sentence compositional process. The main idea in this deep learning algorithm is that the algorithm, in a sense, has a memory of words that came before a current word and thus does better than a simple bag-of-words model that ignores word positions and sequences. As shown in Figure W4, in this algorithm, to characterize sentence sequence, each word is mapped to a vector through a lookup-table (LT) layer, which adds value by staying dynamically tunable based on processed data. For each hidden layer, its input comes from

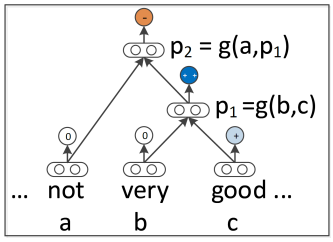
two sources: One is the current word (taken and vectorized through lookup-table layer activations), and the other is the hidden layer's activation one step back in time, which incorporates information from previous phrases (i.e., previous phrase memory). The last hidden layer is considered as the representation of the whole sentence. The example in Figure W4 shows that the three words "not," "very," and "good" are first mapped to a vector through the LT layer. And the last hidden layer h(t) represents the entire (sub)sentence "not very good," to be used for classifying Y, the outcome variable. This algorithm excels in distinguishing negation because it tunes vector representations of sentiment words into valence-polarity-representable ones. Therefore, it shows promising potential dealing with complex sentiment phrases.

Recursive Neural Networks

The second deep learning algorithm is recursive neural networks (Socher et al., 2013). Instead of focusing on sequences as in the recurrent neural networks, the recursive neural networks algorithm focuses on a more complicated sentence *parse-tree* structure that is aware of sentence syntactic context. Intuitively, this algorithm improves on the bag-of-words approach by acknowledging that sentences consist of several syntactic phrases which may vary in sentiment and, when put together to compose a sentence, naturally evolve sentence-level sentiment. This algorithm works to label the sentiment of sentences by labeling the sentiments for each separable syntactic phrase and combining them via recursive neural networks. As shown in Figure W5, in this algorithm, one needs to compute parent phrase representation-vectors in a bottom-up fashion. At the bottom level, the word "not" is classified as neutral (denoted by 0, in white), "very" is classified as neutral, and "good" is classified as (somewhat) positive (denoted by +, in blue). In the middle layer, the phrase "very good" is classified as very positive (denoted by ++, in dark blue). And in the top layer, the entire phrase "not very good" is classified as negative (denoted by -, in orange). From the model perspective, the classifier for the parent node p_1, or phrase "very good," uses a specific and clever compositional function g and node vectors b and c as features. Similarly, the classifier for the top parent node p_2 uses the same composition function g and node vectors a and p_1 as features. Given

this unique composition process, this method can accurately capture the sentiment change (from positive to negative and vice versa) and scope of negation (somewhat negative or very negative). This algorithm can also discern that the sentiment of phrases following the contrastive conjunction "but" in "but not very good" dominates and may be more informative for sentence-level sentiment.

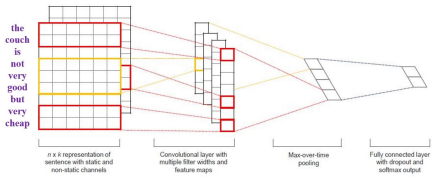
Figure W5: Recursive Neural Networks



From "Recursive deep models for semantic compositionality over a sentiment treebank," by Socher et al., 2015, Proceedings of the conference on empirical methods in natural language processing (EMNLP) vol. 1631, (, 2013), pp. 1642. Copyright 2015 by the Proceedings.com. Reprinted with permission.

Convolutional Neural Networks

Figure W6: Convolutional Neural Networks



From "Convolutional Neural Networks for Sentence Classification," by Kim, Y., 2014 Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25–29, 2014, Doha, Qatar. (2014), pp. 1746–1751. Copyright 2014 by the Proceedings.com. Adapted with permission.

The recursive neural networks algorithm is very powerful, but it requires a parse tree, which is not available in many settings. A parse tree represents the syntactic structure in a sentence using a tree model. For more details, see https://en.wikipedia.org/wiki/Parse_tree. The last algorithm, convolutional neural networks (Kim, 2014, Figure W6), has a data-driven structure that does not rely on externally provided parse trees. This algorithm is similar to the one we presented in Section "Convolutional Neural Networks Framework and Intuition." The key difference is that the outcome variable here is the content dimensions or the associated sentiments instead of conversion.

W4 Full Deep Learning Model

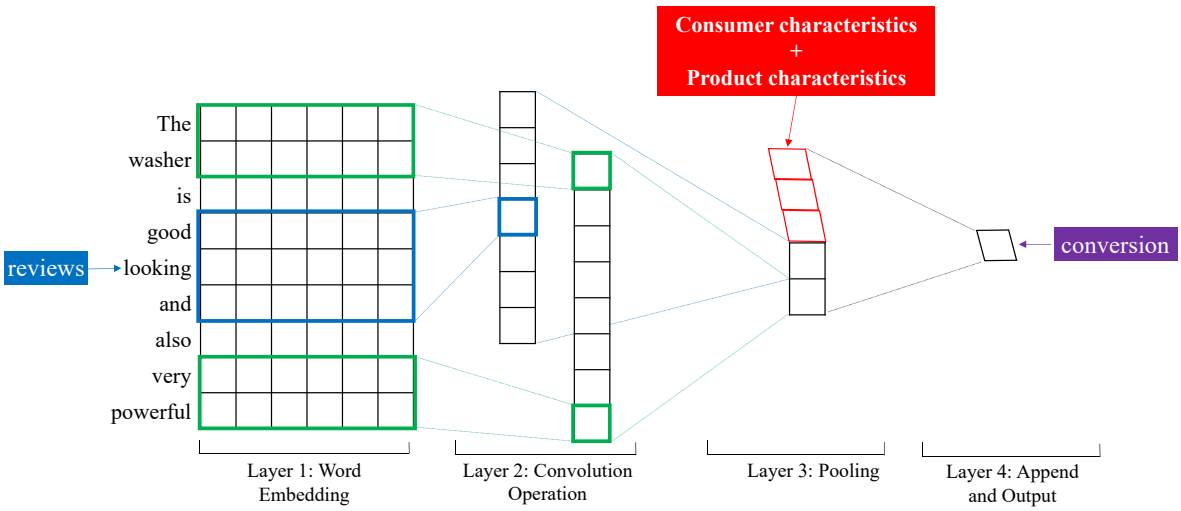
We build a full deep learning model that combines consumer characteristics, product attributes, and review content in a joint framework to predict conversion[2]. We first present the intuition of using a convolutional neural network (CNN) model (Section "Convolutional Neural Networks Framework and Intuition") in our setting and some basic background information about neural network models (Section "Neural Network Basics"). Next we explain the structure (four layers, in Section "CNN Architecture") of the CNN model in detail and the estimation algorithms (Sections "Training: Stochastic Gradient Descent – SGD" and "Regularization: Dropout"). We end (Section "Feature Interpretation") by demonstrating how to interpret the results from the CNN model.

Convolutional Neural Networks Framework and Intuition

Inspired by the works of Kalchbrenner et al. [2014] and Kim [2014], we propose the following convolutional neural network model to examine the impact of review content on sales conversion. The model architecture is illustrated in Figure W7.

[2]The two models have different objectives. The full model is a prediction model, whereas the partial model is a causal inference model. The objective of the full model is to minimize the out-of-sample prediction error. We achieve this by relaxing the functional constraints, using many more features and adding regularization (dropout) to prevent overfitting. In contrast, the objective of the partial model is interpretation and causality. We achieve this by using a theory-driven approach to define six dimensions of content features and by relying on RDIT as the identification strategy. The advantage of the full model is prediction accuracy, but its disadvantage is lack of interpretation and results that can not be used for counterfactual policy simulations. From a managerial perspective, marketers can choose the appropriate model based on their objective. If the goal is to better predict future conversion rate, then the full model should be chosen. On the contrary, if the goal is to predict what would have happened under a different ranking algorithm, such as the one proposed in the section about counterfactuals, then the partial model should be used.

Figure W7: Convolutional Neural Network Model for Conversion



Note. The squares in the figure denote neurons, and the lines denote the connections between inputs and outputs.

Before explaining the details of the model, we first describe the intuition behind convolutional neural network (CNN) natural language processing models. There are two important ideas. The first idea is that some local clues in sentences are more informative for predicting the outcome than others. For example, the exemplar review in Figure W7 says, "the washer is good looking and also very powerful." In this review, the local parts "good looking" and "very powerful" are more informative of the sentiment than the parts "the washer is" and "and also." The second idea behind CNN is that the local clues are informative regardless of their locations in the entire review document. So, if the review changes to "the washer is very powerful and also good looking," then "good looking" and "very powerful" are still the most informative parts.[3] By combining these two ideas, CNN models aim to use "filters" to identify the informative local clues from long sentences and then discard the position information of local clues to reduce the number of parameters in the model and avoid overfitting. Essentially, CNN models use a two-step approach: "convolution"

[3]This second idea behind CNN might not be correct in some settings. For example, the review, "the washer is not very good but very cheap", expresses a positive sentiment, whereas "the washer is very cheap but not very good", expresses a negative sentiment. This kind of more complicated semantic structure is not captured by CNN, and is a big limitation. Other deep learning NLP models, such as recurrent neural networks and recursive neural networks, are able to accommodate more complicated sentence structures. But they have other computational and data limitations that CNN avoids.

and "pooling." "Convolution" applies a filter over each sliding window of the sentence to capture important local clues, whereas "Pooling" aggregates the outputs from the filters by creating a location-insensitive summary statistic. We show more details of convolution in Section "CNN Archtecture."

Neural Network Basics

Above all, the CNN model we propose is a special case of the artificial neural network (ANN) models. The intuition behind ANN is to imitate the way the brain processes information. In an ANN, the basic computation units are called neurons, which are interconnected to form hidden layers, and finally the whole network. Each neuron takes several inputs, multiplies them with the associated weights, sums them, and applies a non-linear activation function to deliver an output (we explain weights and activation functions in Section "Layer 2: Convolution Operation".) Then the output becomes the input of the next layer of neurons. The layers in the neural network reflect information flow.

CNN Architecture

Our model architecture (Figure W7) has four layers. The first layer (the leftmost layer) is the word embeddings (to be explained next) of product reviews, and the second layer is the convolutional layer (described in Section "Layer 2: Convolution Operation"). The third layer is the max-over-time pooling layer (defined in Section "Layer 3: Pooling"), appended with neurons of consumer and product characteristics information. The final layer is our outcome: conversion. The squares in the figure denote neurons, and the lines denote the connections between inputs and outputs.

Layer 1: Word Embedding

In contrast to the sparse representation, content is represented by low-dimensional dense vectors in neural networks. These vectors come from pre-trained word-embeddings. In our implementation,

we use the word2vec embeddings published by Google (https://code.google.com/archive/p/word2vec/). These embeddings are trained on 100 billion words from the Google News dataset using the method created in Mikolov et al. [2013]. In a nutshell, the word2vec model takes the text corpus and transforms it into word vectors while preserving semantic distances between the words as much as possible; thus, each word is represented by a 300-dimensional vector, and the word vectors carry desirable linguistic properties. For example, the vector-distance between similar words is higher than that between dissimilar words.

In Figure W7, the first layer is the word embeddings. Let us use one review as an example. Suppose n is the total number of words in the review and k is the dimensionality of the word embeddings ($k = 300$). We use $\overrightarrow{x}_i \in R^k$, a k-dimensional word vector to represent the ith word in the review, and we use $\overrightarrow{x}_{1:n} = \overrightarrow{x}_1 \oplus \overrightarrow{x}_2 \oplus \ldots \oplus \overrightarrow{x}_n$ to represent the entire review. Here, $\oplus$ is the concatenate operator which stacks all the n-word (column-) vectors to form a $nk \times 1$ vector. In Figure W7, we show a $n \times k$ matrix instead of a $nk \times 1$ vector to make the illustration easier to understand. In the model, we actually create a $nk \times 1$ vector to be further used in equation (1). This vector represents information in the entire review. When there are multiple reviews, they are concatenated in the word embedding in the same order as displayed on the webpage.

Layer 2: Convolution Operation

The next layer is the convolutional layer, which applies the convolution operation or filter to the word embeddings in the first layer. The convolution operation, or filter, is a one-dimensional vector of length h, applied to each sliding window of h words in the sentence. For example, in Figure W7, the green filter size is $h = 2$. The green filter can be written as $\overrightarrow{w} \in R^{hk}$. This is a $hk \times 1$ vector, where h is the window size and k is the dimensionality of the word embeddings in the previous layer ($k = 300$). The filter is first applied to the window of the first two words "the washer," then to the next two words, "washer is," then to the following two words, "is good," and so on. Let i be the current position and $\overrightarrow{x}_{i:i+h-1} \in R^{hk}$ be the window of words or n-grams that the filter is applied to. The output of the convolution operation is

$$c_i = f(\overrightarrow{w} \cdot \overrightarrow{x}_{i:i+h-1} + b), \tag{1}$$

where $\cdot$ denotes inner product, $b \in R$ is the bias parameter, and f is the activation function. The activation function is a non-linear function, which allows neural network models to incorporate non-linear relationships between input variables. We use the rectified linear units (ReLU) as the activation function (Goodfellow et al. 2016, p. 187). The ReLU function is defined as $f(x) = max(x, 0)$. The input to the activation function, $\overrightarrow{w} \cdot \overrightarrow{x}_{i:i+h-1} + b$ is a linear transformation of $\overrightarrow{x}_{i:i+h-1}$. If we use the analogy of a linear regression, we can consider $\overrightarrow{w}$ as the weights and b as the intercept term. Both weights ($\overrightarrow{w}$) and bias (b) are parameters to be estimated. In summary, the convolution operation first applies a linear transformation to the inputs using the weights and bias, and then a non-linear transformation using the activation function.

The filter is rolled over to each sliding window of h words for $i = 1, 2, \ldots$. So, the final output is a vector, $\overrightarrow{c} \in R^{n-h+1}$, called the feature map. Specifically,

$$\overrightarrow{c} = [c_1, c_2, \ldots, c_{n+h-1}].$$

In our setting, we try different window sizes: $h = 2, 3, 4, 5$ to incorporate bi-grams, tri-grams, 4-grams, and 5-grams.

Layer 3: Pooling

The third layer is the pooling layer. The pooling layer applies the max-over-time pooling operation to the feature map created in the previous layer (layer 2). The idea behind the max-over-time pooling is that we want to get the most salient information across all window positions, so

$$\hat{c} = max\{\overrightarrow{c}\}.$$

As mentioned in Section "Convolutional Neural Networks Framework and Intuition," the pooling layer will keep the informative local clues gathered by the filter in layer 2 but discard its position. In other words, $\hat{c}$ ignores which c_i is the biggest, but keeps the maximum value among all c_i's. To sum up, the outcome $\hat{c}$ is the representation of the entire review, and it captures the most indicative information in the review.

Each of the above-mentioned layers 1, 2, 3 extracts one feature from one filter. In reality, we can repeat the process and apply multiple filters with varying window sizes to create multiple features. Let the total number of filters be m. So, the penultimate layer becomes a vector of all the features (each feature corresponds to one filter) extracted from the text data, i.e.,

$$\overrightarrow{d} = [\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_m].$$

For instance, in Figure W7, the green filter ("the washer" and "very powerful") has a window size of 2, whereas the blue filter ("good looking and") uses a window size of 3. After the pooling layer, each filter creates one feature. The two features are concatenated to be passed to the next layer.

In our application, we use a total of 100 filters ($m = 100$) for each window size h ($h = 2,3,4,5$). This results in 400 features, each representing a distinct content dimension. These features are not easily interpretable. In Section "Feature Interpretation," we propose a solution to get around the interpretability issue.

Layer 4: Append and Output

In the last layer, the features extracted from the review text data ($\overrightarrow{d}_{ijt}$) are combined with variables of consumer characteristics ($\overrightarrow{Z_{it}}$, including total products searched and number of used interactions), observed product characteristics ($\overrightarrow{X_{jt}}$ including price, average rating, cumulative number of reviews, percentage of consumers recommended, and number of questions and answers), unobserved product characteristics (ξ_j), and time fixed effects ($Weekend_t$, $Daytime_t$) to predict the final outcome: conversion. The activation function used in the last layer is the softmax function

(Bishop 2006): $softmax(c) = \frac{e^{x_c}}{\sum e^{x_c}}$. In our case, this is equivalent to the logit choice probability function in discrete choice models (Train 2009), because we have a binary outcome of converting versus not converting.

Let y_{ijkt} denote conversion ($y_{ijkt} = 1$ implies purchase, and $y_{ijkt} = 0$ implies no purchase) for consumer i considering product j using device k at time t. Then the specification in the last layer of the neural network is

$$y_{ijkt} = softmax\left(\overrightarrow{w_k^y} \cdot \overrightarrow{d}_{ijt} + b_k^y + \overrightarrow{\theta_k}\overrightarrow{Z_{it}} + \overrightarrow{\gamma_k}\overrightarrow{X_{jt}} + \xi_j + Weekend_t + Daytime_t\right). \quad (2)$$

We append the consumer and product characteristics to text features in the last layer of the neural network to model conversion directly. In other words, this model creates a single, joint deep neural network model to forecast conversion. We call this the "full model." One thing to note is that the text features in this direct approach are not interpretable by face value. This is why, in Section "Partial Deep Learning Approach," we introduce a two-step approach, or a "partial model." Later, we compare the full model and the partial model in terms of their accuracy, efficiency, and interpretability.

Training: Stochastic Gradient Descent – SGD

The estimation algorithm we use is stochastic gradient descent (SGD, see Goodfellow et al. 2016, for more details). It belongs to a broad class of gradient descent (Cauchy, 1847) algorithms. The key advantage of SGD is that it is scalable to a large amount of training data. The basic intuition is that the objective function of the machine learning problem is often the sum or the mean (expectation) of the objective functions applied to each observation. When the number of observations becomes enormous, it's too time consuming to calculate the objective function. So, instead of summing up all the observation-level objective functions, SGD uniformly samples a mini-batch of observations and uses only this subset to approximate the objective function. As a

consequence, the estimation algorithm can speed up without much loss of accuracy. In practice, we use a mini-batch size of 50.

To calculate the gradient, we use the back-propagation method. The idea is that in the neural network model, information flows forward from the input layer to the hidden layers and then finally arrives at the output layer, allowing us to calculate the loss function (objective function). When calculating the gradient of the loss function with respect to a particular parameter (weights and biases in equation (1 and equation (2)), we can reverse this process and allow the information to flow backward through the layers of the network. The gradient calculation applies the chain rule to compute derivatives in each layer until the final derivative is obtained.

Regularization: Dropout

To prevent overfitting, we apply regularization to the model. Specifically, we use the dropout method (Srivastava et al. 2014). Intuitively, this method randomly drops $1-p$ percent of the neurons and keeps only the rest p percent of the neurons in the training process, but in testing, all the neurons will remain for prediction. The mechanism behind this method is model averaging: By dropping subsets of neurons repeatedly, the model creates smaller neural networks, which are then averaged to avoid overfitting. We use a dropout rate of $p = 0.5$ in our practice. The dropout rate is a hyper-parameter. We tune all the hyper-parameters using cross-validation on the development set, a random sample of 10% of the data. Other hyper-parameters, including the number of filters and filter sizes, are chosen in a similar fashion.

Feature Interpretation

Deep learning models are not easily interpretable (Towell and Shavlik 1992). The weights and bias parameters as well as the neurons (including the features created from the filters) in the hidden layers of the model are not meaningful by themselves. To understand what content features affect conversion, we use the approach in Simonyan et al. [2013] to select the most "salient" n-grams.

The intuition is that given the CNN model, we can compute the gradient of each input n-gram ($\overrightarrow{x}_{i:i+h-1}$ in equation (1)) by taking a derivative (using a single back propagation pass through) of the output function with respect to the n-gram. The magnitude of the derivative indicates which n-gram needs to be changed the least to affect the prediction outcome, conversion. For each review, we select the n-gram that has the highest derivative. After obtaining the "salient" n-grams, we run topic modeling (LDA, Blei et al. 2003) to cluster the n-grams to a handful of topics for ease of interpretation. Note that, different from most applications of LDA, in our case, each document contains only one n-gram.

Recall that in Layer 4 (Section "Layer 4: Append and Output") of the CNN model, we append the consumer and product characteristics to the text neurons, and the last activation function is softmax, which resembles a logistic regression. So, the coefficients ($\overrightarrow{\theta_k}$, $\overrightarrow{\gamma_k}$) in equation (2) can be directly interpreted. One caveat is that only the signs of these coefficients can directionally indicate the relationship between consumer/product characteristics (the input variables) and conversion (the output variable). However, because the model does not produce any confidence intervals, we cannot make any valid statistical inference, such as the significance of the coefficients.

Results of the Full Deep Learning Model

Table W1: Coefficients in the Full Deep Learning Models: Consumer and Product Characteristics

	Mobile	PC
# Positive Reviews Read	0.039	0.06
# Negative Reviews Read	-0.088	-0.08
# Products Searched	-0.135	-0.084
# Used Interactions	0.442	0.256
Total # of Reviews	0.004	0.01
%Recommend	0.0007	0.0003
Rating Average	0.57	0.84
Rating Variance	-0.008	-0.001
# Questions	-0.057	0.009
# Answers	0.063	0.051
readability	0.752	-0.066
length	-0.0004	-0.0005
Log_Price	-0.787	-0.868
Obs	156,445	

As explained in Section "Feature Interpretation," the coefficients for the consumer and product characteristics in the full deep learning model can directionally inform us how they affect conversion, though no asymptotic inference can be drawn. In Table W1, we present the model coefficients. We present only a subset of the model coefficients in the last layer. The coefficients for percentage of products recommended, number of questions, number of answers, readability, and length are available upon request.

The signs of the coefficients are consistent with our expectation: price negatively influences conversion. A higher total number of reviews and higher average rating positively affect conversion. Surprisingly, we find a negative effect of the number of products searched in the journey. We believe that this variable indicates the consumer's purchase intention, because if a consumer searches many products, that consumer must have a high willingness to buy. However, counterforces also exist. First, it could be that the consumer is in the early stage of the conversion funnel. Or it could be competition. Recall that our dependent variable is the conversion of the product the reviews are associated with. If the consumer has a large consideration set, the consumer is less likely to purchase any single product because of competition. Thus, the coefficient unveils that the competition effect is stronger than the intention effect. Moreover, the number of used interactions, for example pagination or sorting, is found to have a positive association with conversion because it reflects a higher purchase intention. Expectedly, the number of positive and negative reviews read affects conversion in opposite directions.

Although the coefficients for the text features in the full deep learning model are not directly interpretable, we extract salient n-grams that affect conversion the most using a post-hoc algorithm (see Section "Visualization to Extract Salient Sentences" for details). Table W2 presents the five topics gleaned from running topic modeling (LDA, Blei et al. 2003) on all the salient n-grams. For each topic, we report the top six n-grams with the highest probabilities. Based on the n-grams, we label the five topics as aesthetics, price, feature, favor, and easy-to-use. For example, the topic "aesthetics" is represented by n-grams such as "look good," "lovely looking," and "great color." The topic "price" contains n-grams such as "great value," "the money," "good price," etc. The

topic "feature" represents "what it says," "fit to," "work well," etc. Among all the salient n-grams, "aesthetics" accounts for 36%, while "price" makes up 28%. These are the top two topics that influence conversion. Notice that using this data-driven full deep learning model, we uncover that content related to "easy-to-use" has a big impact on sales. This content dimension attracts consumers but may be bypassed by firms.

Table W2: Topic Modeling of Salient N-grams

Aesthetics 36%	Price 28%	Feature 16%	Favor 12%	Easy-to-use 8%
look good	great value	what it says	highly recommend	easy to
looks good	the price	good quality	love this	to use
lovely looking	for the price	perfect for	good item	to assemble
look really nice	the money	fit to	very happy	to change
feel great	good price	the job	very pleased	set up
great color	good value	work well	quite nice	is fast

Comparison of the Full Deep Learning Model and the Partial Deep Learning Approach

We first compare the prediction performance of the full deep learning model and the partial deep learning model (Section "Partial Deep Learning Approach"). The objective is to directly predict the sales conversion rate in the 1% holdout sample. We compare the models based on three metrics: hit rate (1-misclassification rate), precision, and recall. The results are reported in Table W3. The hit rate for the full model is 88.54%,[4] much higher than the 66.13% for the partial model. This implies that combining all input variables, text features, and consumer/product characteristics in a joint deep learning framework improves prediction accuracy because this is a more efficient way of utilizing information. The two-step approach in the partial model extracts only six dimensions of content dimensions from the reviews, omitting other potentially useful content. We also compare the two deep learning models with alternative models used in the prior literature. Specifically, in Model 3, we use only review rating, volume, and variance (as well as other consumer and product

[4]Hit rate is measured on the balanced sample. Our sample is highly imbalanced because 96% of the journeys result in no conversion. We use the over-sampling approach to construct the balanced sample (Kotsiantis et al. 2006). The hit rates on the original, imbalanced sample are in the range of 96.0% and 98.9%, across various models.

characteristics) but no content features. In Model 4, we add simple content features currently used in the literature, including topics (or LDA, Blei et al. 2003),[5] subjectivity, and readability. In Model 5, we replace the unsupervised LDA model with the supervised seeded LDA model (Jagarlamudi et al. 2012). The comparison reveals that adding the review content features to the models significantly improves model accuracy. This result highlights the importance of studying the impact of review content above and beyond review volume and rating, as well as the simple content features used in the literature.

Table W3: Model Comparison: Full Deep Learning Model vs. Partial Deep Learning Approach

	DV: Conversion	Balanced			Imbalanced		
		Hit Rate	Precision	Recall	Hit Rate	Precision	Recall
1	Full Deep Learning Model	88.54%	83.36%	96.48%	98.90%	20.08%	0.03%
2	Partial Deep Learning Approach	66.13%	63.43%	79.06%	97.23%	8.77%	2.35%
3	No Content (but with rating/volume/variance)	57.51%	59.13%	50.96%	96.02%	5.86%	1.50%
4	Simple Content Features (with rating/volume/variance/ topics /subjectivity/readability)	58.13%	56.90%	69.58%	96.28%	5.00%	1.66%
5	Simple Content Features (Seeded LDA)	58.05%	58.02%	60.55%	96.23%	5.81%	1.63%

Note: All models are homogeneous.

W5 Salience Extraction Method

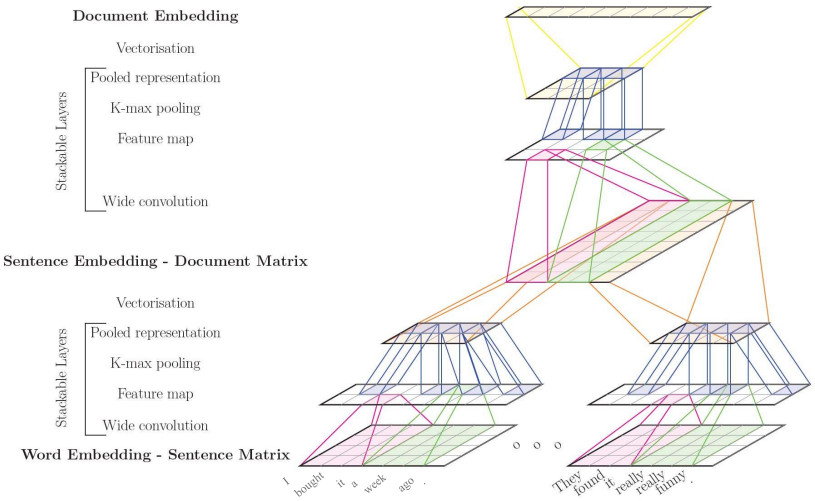
We implement a post-hoc method created by Denil et al. [2014] that adapts visualization techniques from computer vision to automatically extract relevant sentences from labeled text data. In essence, it is a CNN that has a hierarchical structure divided into a sentence level and a document level.

At the sentence level, the algorithm transforms embeddings for the words in each sentence into an embedding for the entire sentence. At the document level, another CNN transforms sentence embeddings from the first level into a single embedding vector that represents the entire document.

[5]We use the distribution over topics estimated using LDA. We choose the number of topics as 20 based on Perplexity [Blei et al., 2003].

Figure W8 is a schematic of the algorithm. Specifically, at the bottom layer, word embeddings are concatenated into columns to form a sentence matrix. For example, each word in the sentence "I bought it a week ago" is represented by a vector of length 5. Then these vectors are concatenated to form a $7*5$ dimensional sentence matrix (7 denotes the number of words in the sentence, including punctuation). The sentence level CNN applies a cascade of operations (convolution, pooling, and nonlinearity) to transform the projected sentence matrix into an embedding for the sentence. The sentence embeddings are then concatenated into columns to form a document matrix (the middle layer in the figure). In the example, the sentence embeddings for the first sentence, "I bought it a week ago," until the last sentence, "They found it really really funny," are concatenated to form the document matrix. The document model then applies its own cascade of operations (convolution, pooling, and nonlinearity) to form an embedding for the whole document, which is fed into a final layer (softmax) for classification. After this algorithm is trained, it can then be used to extract salient sentences by identifying sentence locations that have the highest amount of influence to loss function. The first step in the extraction procedure is to create a saliency map for the document by assigning an importance score to each sentence. These saliency scores are calculated using gradient magnitudes, because the derivative indicates which words need to be changed the least to affect the score the most. The following step is to rank sentences based on the saliency score and highlight the sentences with the highest score.

Figure W8: Using Convolutional Neural Networks to Extract Salient Sentences



From "Extraction of Salient Sentences from Labelled Documents," by Denil et al., 2014. Copyright 2014 by University of Oxford. Adapted with permission.

W6 Summary Statistics of Variables in the Regressions

Our data consist of seven tables: four user behavior tables and three domain tables. The four user behavior tables are pageview, review impression, usedfeature,[6] and transaction, as described in Section "Descriptions of Consumers' Review-Reading Behaviors." The three domain tables store the information of product reviews, questions, and answers. Here is the description of each table.

Table W4: Data Structure

Table Name	Description
pageview	a single product or category page view for a customer
impression	a single exposure to a product review
usedfeature	a single user engagement with a product review
transaction	a single product purchase made by a customer
review	a single review of a product
question	a single question related to a product
answer	a single answer related to a product

All the seven tables are used to construct the consumer decision journey. The four behavioral tables are linked to the journey using the key variable "userid." The three domain tables are linked

[6]Referred to as used-interaction in Section "Descriptions of Consumers' Review-Reading Behaviors."

to the journey using the "productid" variable. After constructing the journeys, we can calculate the variables used in the regression equation (4) for each journey.

W7 Survey Instrument to Content-Code Review Content

Figure W9: **Survey Instrument Shown to Amazon Mechanical Turkers for Identifying Review Contents**

CONTENT DESCRIPTION

1. Price: Any content regarding the price of the item that's under review. The consumer may find the price too high, too low, or just right.

Example reviews with price content:
a. "This overpriced junk broke after using twice!"
b. "Fair price given it was less than 20 dollars."

2. Performance and Feature: This dimension involves observable and measurable attributes of the product. These include the products' primary characteristics that can be measured and compared. For example, if the product is an iPhone, the performance and feature attributes would include topics like screen size, weight, image and video resolution, camera megapixel, etc. If the product is a curtain, the performance could include topics about fabric feelings, size, laundry requirement, thickness, whether it blocks light etc.

Example reviews with performance and feature content:
a. "The screen size is quite small at 3.5 inches"
b. "The Aveeno Lotion's smell was great"

3. Reliability and Durability: Reliability reflects the probability of a product malfunctioning or failing to work as per the customers' satisfaction. For example, if a customer purchases a camera and finds operating defects within a short period of time, the product is ranked lower on the reliability. Durability measures the product life. Products may have high durability or lifespan (e.g., well built camera lens) or have low durability and lifespan (e.g., poorly made camera lenses which are fragile).

Example reviews with reliability and durability content:
a. "These earbuds broke after 3 months of regular usage"
b. "These new nokia phones are built like bricks! After we are gone, nokia will remain"

4. Conformance: This dimension reflects the degree to which the product's design and operating characteristics meet established standards. This dimension is perceived as the amount of divergence of the product feature specifications from an ideal or accepted standard. For example, if an automobile promises noise-free operation, but customers find that the car is actually quite noisy, then they would rank the car low on conformance.

Example reviews with conformance content:
a. "The product does exactly what they says it would do...hydrating my dry skin."
b. "The jacket wasn't rainproof as advertised!"

5. Aesthetics: This is a subjective measure. The aesthetic dimension captures how a product looks, feels, sounds, tastes, or smells, and is clearly a matter of personal judgment and a reflection of individual preference. For example, a person using an iPhone might feel that the phone has a "decent look and feel." This purely reflects the customer's own aesthetic preferences, as other customers might have differing opinions on what a "decent" look and feel might entail.

Example reviews with aesthetics content:
a. "The lamp's sleek appearance is pleasing and I got many complements."
b. "The color of the jean was not what I was looking for. It looks so cheap!"

6. Perceived Quality: Consumers do not always have complete information about a product or service's attributes, and hence, indirect measures may be their only basis for comparing brands. A leading source of perceived product quality is reputation of the brand. For example, consumers might prefer a new line of shoes purely because it comes from a leading shoe manufacturer that has a proven record of good quality e.g. Nike, Adidas etc.

Example reviews with perceived quality content:
a. "Have been using HP ink for 5 yrs and think it's the best on the market!"
b. "What's up with Samsung lately? The TVs are overpriced for what they offer"!

QUESTIONS

1. ***[Price]*** This review contains content regarding **pricing of product**

   YES/NO
   If you answered yes above, judge if the sentiment regarding this specific content is negative or positive. If answered no, then select Not Applicable.
   Sentiment in Likert Scale 1 (Strongly Negative) 7 (Strongly Positive)
   We exclude identical answer parts for other questions for brevity

2. ***[Performance and Feature]*** This review talks about **presence or absence of product features and performances**
3. ***[Reliability and Durability]*** This review describes the experience with the **durability or reliability or product malfunctioning or failing** to work as per the customer's satisfaction.
4. ***[Conformance]*** This review **compares** the performance of the product **with pre-existing standards or set expectations or as advertised.**
5. ***[Aesthetics]*** This review talks about **how a product looks, feels, sounds, tastes, or smells,** and is clearly a matter of personal judgment and a reflection of individual preference.
6. ***[Perceived quality]*** This review talks about **indirect measures of the quality of the product** like the reputation of the product brand or based on a history of past purchases.

W8 Amazon Mechanical Turk Strategies and Cronbach's Alpha

Following best-practices in the literature, we employ the following strategies to improve the quality of classification by the Turkers in our study.

1. For each message, at least 5 different Turkers' inputs were recorded. We obtained the final classification by a majority-voting rule.

2. We restricted the quality of Turkers included in our study to only those with at least 100 reported completed tasks and 97% or better reported task-approval rates.

3. We used only Turkers from the US to filter out those potentially not proficient in English and to closely match the user-base from our data (recall, our data has been filtered to include only pages located in the US).

4. We created a sample test, and only those who passed this test, in addition to possessing the above qualifications, were allowed to work.

5. We refined our survey instrument through an iterative series of about 10 pilot studies, in which we asked Turkers to identify confusing or unclear questions. In each iteration, we asked 10–30 Turkers to identify confusing questions and the reasons they found those questions confusing. We refined the survey in this manner until almost all queried Turkers stated that no questions were confusing.

6. To filter out participants who were not paying attention, we included an attention question that asked the Turkers to click a certain input. Responses from Turkers who failed the verification test were dropped from the data.

7. On average, we found that review tagging took about 4 minutes, and it typically took at least 30 seconds or more to completely read the tagging questions. We defined less than 30

seconds to be too short, and discarded any review tags with completion times shorter than that duration, to filter out inattentive Turkers and automated programs ("bots").

8. Once a Turker tagged more than 20 messages, a couple of tagged samples were randomly picked and manually examined for quality and performance. This process identified dozens of high-volume Turkers who completed all surveys with seemingly random answers but managed to pass time-filtering requirements. We concluded that these were automated programs. These results were dropped, and the Turkers were "hard blocked" from the survey, via the blocking option provided in AMT.

Figure W10: **Cronbach's Alphas for 5,000 Reviews**

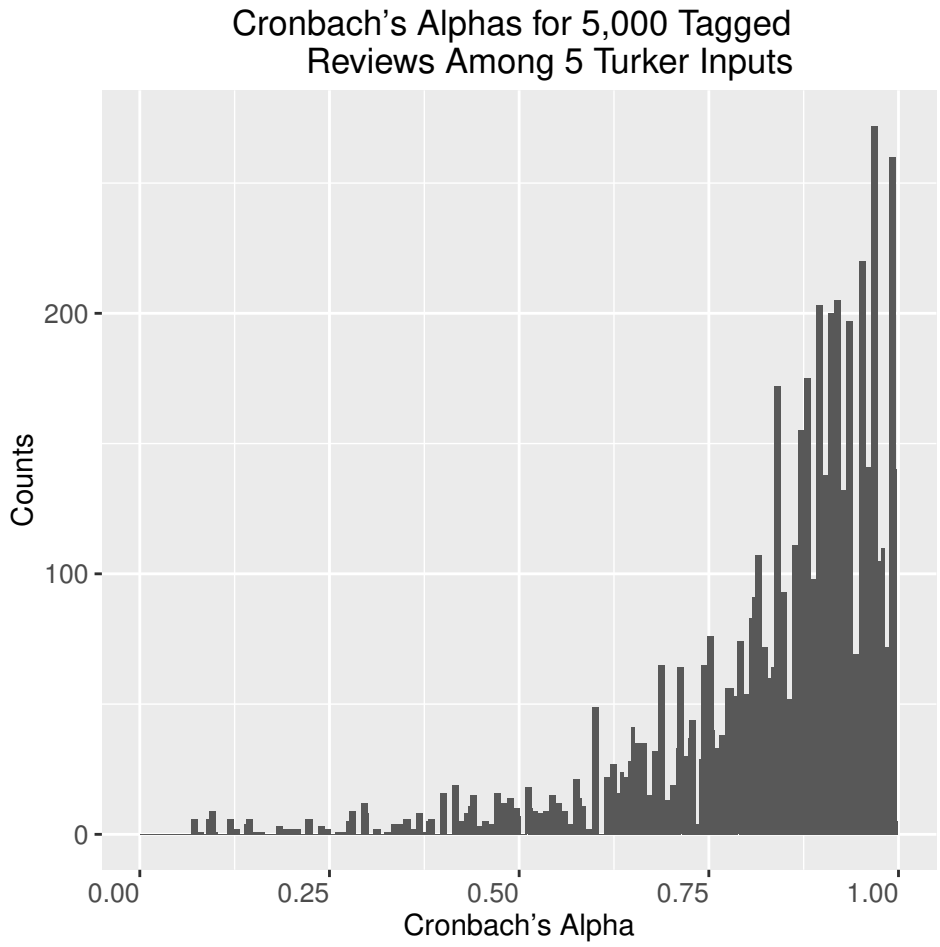


Figure W10 presents the histogram of Cronbach's alphas, a commonly used inter-rater reliability

measure, obtained for 5,000 reviews. The average Cronbach's alpha for our tagged reviews is 0.84 (median 0.88), well above the typically acceptable thresholds of 0.7. About 84% of the reviews obtained an alpha higher than 0.7, and 90% obtained an alpha higher than 0.6. For robustness, we replicated the study with only those messages with alphas above 0.7 (4,193 messages) and found that our results were qualitatively similar.

W9 Model Comparison for Information Detection

Table W5: Model Comparison: Information Detection

Classifier/Accuracy %		SVM + BoW	NB+ BoW	Recurrent-LSTM	Convolutional
Mean	Precision	0.358	0.484	0.496	0.955
	Recall	0.098	0.563	0.483	0.946
	F1	0.151	0.472	0.475	0.950
Aesthetics	Precision	0.304	0.531	0.492	0.917
	Recall	0.057	0.623	0.516	0.863
	F1	0.097	0.574	0.504	0.889
Conformance	Precision	0.211	0.134	0.235	0.971
	Recall	0.061	0.455	0.162	0.992
	F1	0.094	0.207	0.192	0.981
Durability	Precision	0.350	0.333	0.373	0.910
	Recall	0.064	0.550	0.648	0.921
	F1	0.109	0.415	0.473	0.916
Feature	Precision	0.922	0.966	0.921	0.999
	Recall	0.327	0.570	0.791	0.951
	F1	0.482	0.717	0.851	0.974
Brand	Precision	0.059	0.082	0.094	0.983
	Recall	0.026	0.474	0.176	1.000
	F1	0.036	0.140	0.122	0.992
Price	Precision	0.304	0.858	0.861	0.948
	Recall	0.051	0.708	0.604	0.952
	F1	0.088	0.776	0.710	0.950

Note that Socher et al. [2013] is suitable only for sentiment analysis, so we could not perform information detection for the recursive neural networks model.

W10 Hierarchical Bayes Model

Let the total number of product categories be C and the total number of coefficients be NIV. If we concatenate the coefficients in equation (4) to a $C \times NIV$ matrix B where each row is a vector of coefficients associated with a particular category, i.e., $B_c = \left[\overrightarrow{\tau_{kc}}, \delta_{nkc}, \overrightarrow{\theta_{kc}}, \overrightarrow{\gamma_{kc}}, \xi_c, Weekend_{tc}, Daytime_{tc}\right]$, then the multivariate regression takes the form

$$\underbrace{B}_{C\times NIV} = \underbrace{\Delta}_{C\times NIV} + U. \tag{3}$$
$$where\, U \sim N\left(0, V_\beta\right)$$

The $C \times NIV$ matrix Δ contains the mean value of the NIV coefficients for each category. The error term U is assumed to be distributed normally with mean zero and covariance matrix V_β. To complete the model formulation, we set the following priors for Δ and V_β:

$$vec\left(\Delta|V_\beta\right) \sim N\left(vec\left(\overline{\Delta}\right), V_\beta \otimes A^{-1}\right)$$
$$V_\beta \sim IW\left(v, V\right).$$

They are the natural conjugate priors for the multivariate regression in equation (3).

W11 Endogenous Review-Reading Behaviors

In reality, consumers endogenously decide which reviews to read and how many reviews to read. We do not model this endogenous process explicitly. However, the endogenous review-reading behavior will not affect the counterfactual result for the following reasons.

First of all, regarding which reviews to read, our assumption is that consumers use a top down fashion, i.e., reading reviews from the top of the page to the bottom of the page. This assumption is supported by previous eye tracking studies. The retailer in our study posts reviews in the reverse chronological order. Therefore, consumers read the most recent reviews before the old reviews. Importantly, consumers can also engage in "used-interaction;" i.e., this online retailer allows users to filter reviews by star ratings. For example, a consumer can filter to look at only one star reviews. When doing so, all the one star reviews are displayed to the consumer, also in reverse chronological order. Because the order is always reverse chronological, no matter whether a consumer filters or not, our identification is immune to consumers' filtering/sorting behavior. In other words, if a consumer in our data used the filter to select only one star reviews, in the counterfactual, we also let this consumer read only one star reviews. Even though the star rating remains unchanged, we can change the content of reviews read by consumers in the counterfactual. In the observed data, a consumer may read only a one star review about "conformance." In the counterfactual, we re-order the reviews by content, so this consumer will read a one star review about "aesthetics" instead. Because the counterfactual does not alter consumers' review-reading behavior with respect to "which reviews to read," this problem is eliminated.

Second, regarding how many reviews to read, we provide additional evidence that, in our data, the content of the first review is uncorrelated with the number of reviews read. In other words, we find that no matter whether a consumer read a review about "conformance" at the beginning or a review about "aesthetics" at the beginning, he will read the same number of reviews in both scenarios. We test this hypothesis-whether the number of reviews read is uncorrelated with the content of the first review, because this is what we did in the counterfactual. On page 28, we state that "We implement a counterfactual scenario where for each product, we randomly select an associated review that contains positive aesthetics information and move it from a lower position to the first position in the set of reviews read by each consumer." We acknowledge that if we did other counterfactuals, we would probably need to more carefully model the endogenous review-reading behavior. Now we present the data evidence. In Table W6 below, we show the distribution of

"the number of reviews read" by the six content dimensions: aesthetics, conformance, durability, feature, brand, and price. For the distribution statistics, we present number of observations (N, column 2), mean, standard deviation, and median of "the number of reviews read."

Table W6: Distribution of Number of Reviews Read by Content in the First Review Read

	N	Mean	Std	Median
Aesthetics	25105	11.1265485	7.74282639	10
Conformance	9275	11.1926685	7.35463469	10
Durability	18521	11.007559	7.46181403	10
Feature	66011	11.1631243	7.65780554	10
Brand	2844	11.4388186	7.58973413	10
Price	28746	11.2147081	7.57472158	10

The table shows that, the mean, standard deviation, and median of the "number of reviews read" are all very similar across scenarios. We also perform a two sample Kolmogorov–Smirnov test to see whether the samples are drawn from the same distribution under different scenarios. The table below shows the p values of the pair-wise Kolmogorov–Smirnov tests. As you can see, across the fifteen pairs, only two are significantly different from each other. This confirms our hypothesis that the number of reviews read is uncorrelated with the content of the first review.

Table W7: P_values of the Pair-wise Kolmogorov–Smirnov Tests

	Aesthetics	Conformance	Durability	Feature	Brand	Price
Aesthetics	X	X	X	X	X	X
Conformance	0.0759	X	X	X	X	X
Durability	0.3313	0.0533	X	X	X	X
Feature	0.1450	0.4484	0.0334	X	X	X
Brand	0.3435	0.8766	0.0584	0.4784	X	X
Price	0.0076	0.9686	0.0030	0.4842	0.4170	X

Therefore, although our model does not explicitly account for the endogenous review-reading behavior, after carefully checking the data evidence, we find that this does not limit our capability to answer the counterfactual question: how would re-ordering the reviews affect sales conversion?

W12 Unit of Analysis

The unit of analysis in the regression is journey-product. We use only the type 4 and type 5 journeys in the regression. In fact, we only use a subset of the type 4 and type 5 journeys in which the focal products had a least one new review posted during the sample period. We have to narrow down to this subset for the purpose of identification. The intuition is that for a focal product, if no new review was posted during the sample period, then there is no variation in the review content read by consumers, so we cannot identify the effect of review content on sales. This subset leaves us with 58282 journeys. Importantly, our unit of analysis is not a journey, but a journey and product combination. In some journeys, the consumer read reviews of multiple products (an average of 2.7) before purchasing, so we looked at the impact of reviews for each product separately. This explains why we have 156,445 observations in the regression.

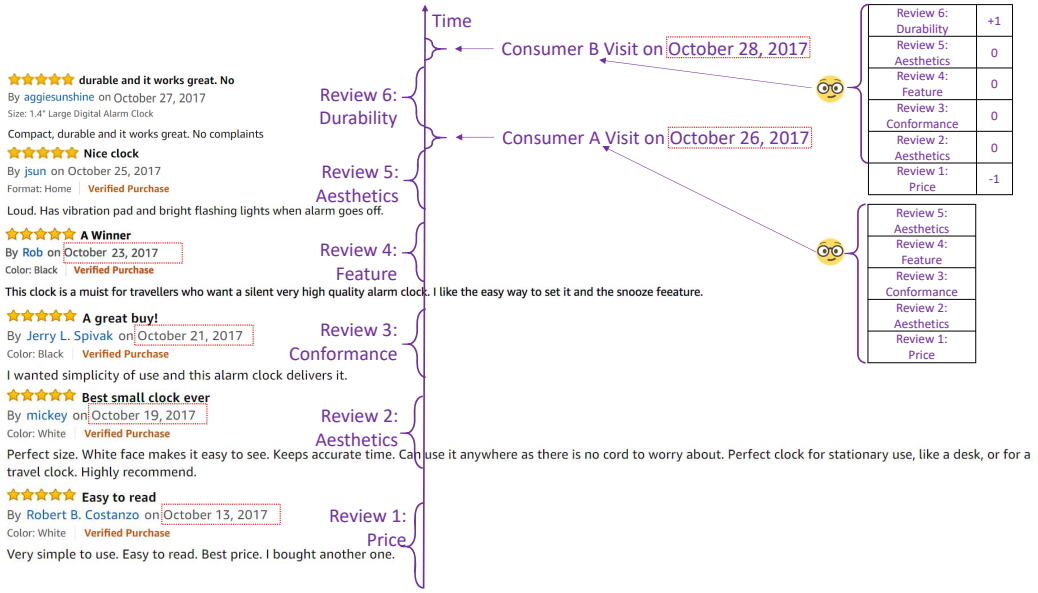
We now explain why we choose the unit of analysis at the journey-product level. Our assumption is that the reviews a consumer read affects the purchase likelihood only for the focal product but not for the other products considered in the journey. Specifically, if the review content for product A is favorable, then the likelihood of purchasing product A will increase. But the review content of product A won't affect the purchase likelihood of product B, which is also in the journey. This specification rules out the competition effect. As you can see from the conversion rate formula on page 17 ($ConversionRate_{ijkt} = (exp(u_{ijkt}))/(1+exp(u_{ijkt}))$), this is a binomial logit function instead of a multinomial logit function. We have two justifications for this binomial logit specification. First of all, in a robustness check, we also tested the multinomial specification; the signs of the coefficients are consistent with the current specification, but some coefficients became insignificant due to the smaller sample size. Note that the sample size of the multinomial specification is equal to the number of journeys, which is about 1/3 of the number of observations in the current specification. Essentially, allowing for competition dramatically reduces the number of observations in the regression, which reduces the power of the analysis. However, the insights we obtain from the two specifications are very consistent. Therefore, we make a tradeoff to use the bino-

mial logit function. Second, the binomial logit function requires a much shorter computing time because we do not have to pool the reviews for all the products in the CNN model. The pooling makes the convolutional operator in the second layer of CNN much slower. So above all, due to the concerns for economic insights, statistical power, and computational burden, we choose to use journey-product as the unit of analysis instead of journey itself.

W13 RDiT Example

We want to clarify that no matter if a consumer reads 5 reviews or 3 reviews, RDiT can be applied. Please see the illustration below. When review #6 forces review #1 to the second page, the content dimensions in review #6 (durability in the figure) will be assigned as a positive treatment (+1), whereas the content dimensions in review #1 (price) will be assigned as a negative treatment (-1). This is because consumer A reads reviews #1 to #5, whereas consumer B reads reviews #2 to #6. Therefore, consumer B receives more content information about durability in review #6 and less information about price in review #1 compared to consumer A.

Figure W11: Illustration of Multiple Treatments



References

Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Augustin Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

Misha Denil, Alban Demiraj, and Nando de Freitas. Extraction of salient sentences from labelled documents. Technical report, University of Oxford, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

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

Jagadeesh Jagarlamudi, Hal Daumé III, and Raghavendra Udupa. Incorporating lexical priors into topic models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 204–213. Association for Computational Linguistics, 2012.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, 2014.

Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

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

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, volume 1631, page 1642, 2013.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Geoffrey Towell and Jude W Shavlik. Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. In *Advances in neural information processing systems*, pages 977–984, 1992.

Kenneth E Train. *Discrete choice methods with simulation*. Cambridge university press, 2009.