CAPTURING MEANING IN PRODUCT REVIEWS WITH CHARACTER-LEVEL GENERATIVE TEXT MODELS

Zachary C. Lipton *
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093, USA
zlipton@cs.ucsd.edu

Sharad Vikram †
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093, USA
svikram@cs.ucsd.edu

Julian McAuley ‡
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093, USA
jmcauley@cs.ucsd.edu

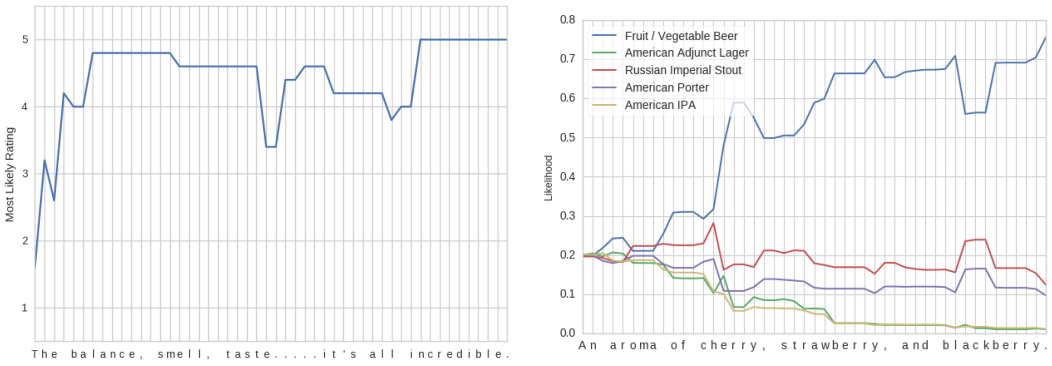
ABSTRACT

We present a character-level recurrent neural network that generates relevant and coherent text given auxiliary information such as a sentiment or topic.[1] Using a simple input replication strategy, we preserve the signal of auxiliary input across wider sequence intervals than can feasibly be trained by back-propagation through time. Our main results center on a large corpus of 1.5 million beer reviews from BeerAdvocate. In generative mode, our network produces reviews on command, tailored to a star rating or item category. The generative model can also run *in reverse*, performing classification with surprising accuracy. Performance of the *reverse* model provides a straightforward way to determine what the generative model *knows* without relying too heavily on subjective analysis. Given a review, the model can accurately determine the corresponding rating and infer the beer's category (IPA, Stout, etc.). We exploit this capability, tracking perceived sentiment and class membership as each character in a review is processed. Quantitative and qualitative empirical evaluations demonstrate that the model captures meaning and learns nonlinear dynamics in text, such as the effect of negation on sentiment, despite possessing no *a priori* notion of words. Because the model operates at the character level, it handles misspellings, slang, and large vocabularies without any machinery explicitly dedicated to the purpose.

1 INTRODUCTION

Our work is motivated by an interest in product recommendation. Currently, recommender systems assist users in navigating an unprecedented selection of items, personalizing services to a diverse set of users with distinct individual tastes. Typical approaches surface items that a customer is likely to purchase or rate highly, providing a basic set of primitives for building functioning internet applications. Our goal is to create richer user experiences, not only recommending products but generating descriptive text. For example, engaged users may wish to know what precisely their impression of an item is expected to be, not simply whether the item will warrant a thumbs up or thumbs down. *Consumer reviews* can address this issue to some extent, but large volumes of reviews are difficult to sift through, especially if a user is interested in some niche aspect. Our fundamental goal is to resolve this issue by building systems that can both generate contextually appropriate descriptions and infer items from abstract descriptions.

*Author website: http://zacklipton.com
†Author website: http://www.sharadvikram.com
‡Author website: http://cseweb.ucsd.edu/∼jmcauley/
[1] Live web demonstration of rating and category-based review generation (http://deepx.ucsd.edu/beermind)



(a) "The balance, smell, taste...it's all incredible."

(b) "An aroma of cherry, strawberry, and blackberry."

Figure 1: Our generative model runs *in reverse*, inferring ratings and categories given reviews without any *a priori* notion of words.

Character-level Recurrent Neural Networks (RNNs) have a remarkable ability to generate coherent text (Sutskever et al., 2011), appearing to hallucinate passages that plausibly resemble a training corpus. In contrast to word-level models, they do not suffer from computational costs that scale with the size of the input or output vocabularies. This property is alluring, as product reviews draw upon an enormous vocabulary. Our work focuses on reviews scraped from Beer Advocate (McAuley and Leskovec, 2013). This corpus contains over 60,000 distinct product names alone, in addition to standard vocabulary, slang, jargon, punctuation, and misspellings.

Character-level LSTMs powerfully demonstrate the ability of RNNs to model sequences on multiple time scales simultaneously, i.e., they learn to form words, to form sentences, to generate paragraphs of appropriate length, etc. To our knowledge, all previous character-level generative models are unsupervised. However, our goal is to generate character-level text in a supervised fashion, conditioning upon *auxiliary input* such as an item's rating or category[2]. Such conditioning of sequential output has been performed successfully with word-level models, for tasks including machine translation (Sutskever et al., 2014), image captioning (Vinyals et al., 2015; Karpathy and Fei-Fei, 2014; Mao et al., 2014), and even video captioning (Venugopalan et al., 2014). However, despite the aforementioned virtues of character-level models, no prior work, to our knowledge, has successfully trained them in such a supervised fashion.

Most supervised approaches to word-level generative text models follow the encoder-decoder approach popularized by Sutskever et al. (2014). Some auxiliary input, which might be a sentence or an image, is encoded by an encoder model as a fixed-length vector. This vector becomes the initial input to a decoder model, which then outputs at each sequence step a probability distribution predicting the next word. During training, weights are updated to give high likelihood to the sequences encountered in the training data. When generating output, words are sampled from each predicted distribution and passed as input at the subsequent sequence step. This approach successfully produces coherent and relevant sentences, but is generally limited to generating sentences (e.g. typically less than 10 words in length), as the model gradually 'forgets' the auxiliary input.

However, to model longer passages of text (such as reviews), and to do so at the character level, we must produce much longer sequences than seem practically trainable with an encoder-decoder approach. To overcome these challenges, we present an alternative modeling strategy. At each sequence step t, we concatenate the auxiliary input vector $\boldsymbol{x}_{aux}$ with the character representation $\boldsymbol{x}_{char}^{(t)}$, using the resulting vector $\boldsymbol{x'}^{(t)}$ to train an otherwise standard generative RNN model. It might seem redundant to replicate $\boldsymbol{x}_{aux}$ at each sequence step, but by providing it, we eliminate pressure on the model to memorize it. Instead, all computation can focus on modeling the text and its interaction with the auxiliary input.

In this paper, we implement the concatenated input model, demonstrating its efficacy at both review generation and traditional supervised learning tasks. In generative mode, our model produces

[2]We use *auxiliary input* to differentiate the "context" input from the character representation passed in at each sequence step. By supervised, we mean the output sequence depends upon some auxiliary input.

convincing reviews, tailored to a star rating and category. We present a live web demonstration of this capability (http://deepx.ucsd.edu/beermind). This generative model can also run *in reverse*, performing classification with surprising accuracy (Figure 1). The purpose of this model is to generate text, but we find that classification accuracy of the reverse model provides an objective way to assess what the model has learned. An empirical evaluation shows that our model can accurately classify previously unseen reviews as positive or negative and determine which of 5 beer categories is being described, despite operating at the character level and not being optimized directly to minimize classification error. Our exploratory analysis also reveals that the model implicitly learns a large vocabulary and can effectively model nonlinear dynamics, like the effect of negation. Plotting the inferred rating as each character is encountered for many sentences (Figure 1) shows that the model infers ratings quickly and anticipates words after *reading* particularly informative characters.

2 The Beer Advocate Dataset

We focus on data scraped from Beer Advocate as originally collected and described by McAuley and Leskovec (2013). Beer Advocate is a large online review community boasting 1,586,614 reviews of 66,051 distinct items composed by 33,387 users. Each review is accompanied by a number of numerical ratings, corresponding to "appearance", "aroma", "palate", "taste", and also the user's "overall" impression. The reviews are also annotated with the item's category. For our experiments on ratings-based generation and classification, we select 250,000 reviews for training, focusing on the most active users and popular items. For our experiments focusing on generating reviews conditioned on item category, we select a subset of 150,000 reviews, 30,000 each from 5 of the top categories, namely "American IPA", "Russian Imperial Stout", "American Porter", "Fruit/Vegetable Beer", and "American Adjunct Lager". From both datasets, we hold out 10% of reviews for testing.

3 Recurrent Neural Network Methodology

Recurrent neural networks extend the capabilities of feed-forward networks to handle sequential data. Inputs $\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(T)}$ are passed to the network one by one. At each step t, the network updates its hidden state as a function of both the current input and the previous step's hidden state, outputting a prediction $\hat{\boldsymbol{y}}^{(t)}$. In this paper, we use RNNs containing long short term memory (LSTM) cells introduced by Hochreiter and Schmidhuber (1997) with forget gates introduced in Gers et al. (2000), owing to their empirical successes and demonstrated ability to overcome the exploding/vanishing gradient problems suffered by other RNNs (Bengio et al., 1994). In short, each memory cell has an internal state s in which activation is preserved along a self-connected recurrent edge. Each cell also contains three sigmoidal gating units for input (i), output (o), and to forget (f) that respectively determine when to let activation into the internal state, when to pass activation to the rest of the network, and when to flush the cell's hidden state. The output of each LSTM layer is another sequence, allowing us to stack several layers of LSTMs as in Graves (2013). At step t, each LSTM layer $\boldsymbol{h}_l^{(t)}$ receives input from the previous layer $\boldsymbol{h}_{l-1}^{(t)}$ at the same sequence step and the same layer at the previous time step $\boldsymbol{h}_l^{(t-1)}$. The recursion ends with $\boldsymbol{h}_0^{(t)} = \boldsymbol{x}^{(t)}$ and $\boldsymbol{h}_l^{(0)} = \mathbf{0}$. Formally, for a layer $\boldsymbol{h}_l$ the equations to calculate the forward pass through an LSTM layer are:

$$\boldsymbol{g}_l^{(t)} = \phi(W_l^{\mathrm{gx}} \boldsymbol{h}_{l-1}^{(t)} + W_l^{\mathrm{gh}} \boldsymbol{h}_l^{(t-1)} + \boldsymbol{b}_l^{\mathrm{g}})$$

$$\boldsymbol{i}_l^{(t)} = \sigma(W_l^{\mathrm{ix}} \boldsymbol{h}_{l-1}^{(t)} + W_l^{\mathrm{ih}} \boldsymbol{h}_l^{(t-1)} + \boldsymbol{b}_l^{\mathrm{i}})$$

$$\boldsymbol{f}_l^{(t)} = \sigma(W_l^{\mathrm{fx}} \boldsymbol{h}_{l-1}^{(t)} + W_l^{\mathrm{fh}} \boldsymbol{h}_l^{(t-1)} + \boldsymbol{b}_l^{\mathrm{f}})$$

$$\boldsymbol{o}_l^{(t)} = \sigma(W_l^{\mathrm{ox}} \boldsymbol{h}_{l-1}^{(t)} + W_l^{\mathrm{oh}} \boldsymbol{h}_l^{(t-1)} + \boldsymbol{b}_l^{\mathrm{o}})$$

$$\boldsymbol{s}_l^{(t)} = \boldsymbol{g}_l^{(t)} \odot \boldsymbol{i}_l^{(i)} + \boldsymbol{s}_l^{(t-1)} \odot \boldsymbol{f}_l^{(t)})$$

$$\boldsymbol{h}_l^{(t)} = \phi(\boldsymbol{s}_l^{(t)}) \odot \boldsymbol{o}_l^{(t)}.$$

Here, σ denotes an element-wise sigmoid function, ϕ an element-wise $tanh$, and $\odot$ is an element-wise product. While a thorough treatment of the LSTM is beyond the scope of this paper, we refer to our review of the literature (Lipton et al., 2015) for a gentler unpacking of the material.

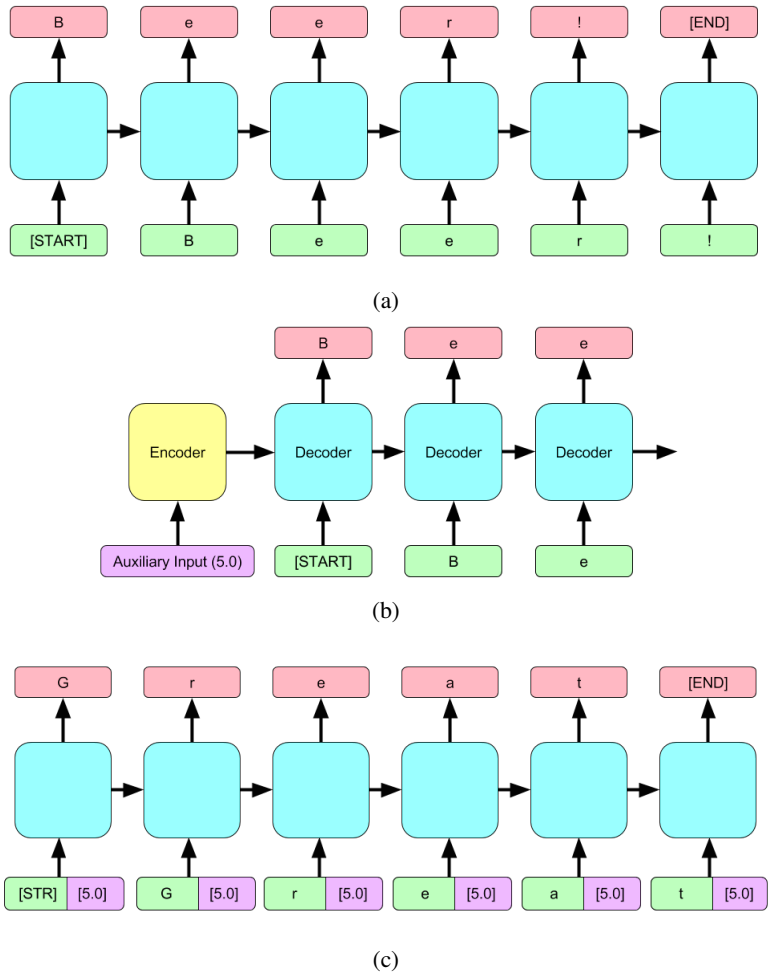


Figure 2: (a) Standard generative RNN; (b) encoder-decoder RNN; (c) concatenated input RNN.

3.1 Generative Recurrent Neural Networks

Before introducing our contributions, we review the generative RNN model of Sutskever et al. (2011; 2014) on which we build. A generative RNN is trained to predict the next token in a sequence, i.e. $\hat{\boldsymbol{y}}^t = \boldsymbol{x}^{(t+1)}$, given all inputs to that point $(\boldsymbol{x}^1, ..., \boldsymbol{x}^t)$. Thus input and output strings are equivalent but for a one token shift (Figure 2a). The output layer is fully connected with softmax activation, ensuring that outputs specify a distribution. Cross entropy is the loss function during training.

Once trained, the model is run in generative mode by sampling stochastically from the distribution output at each sequence step, given some starting token and state. Passing the sampled output as the subsequent input, we generate another output conditioned on the first prediction, and can continue in this manner to produce arbitrarily long sequences. Sampling can be done directly according to softmax outputs, but it is also common to *sharpen* the distribution by setting a *temperature* ≤ 1, analogous to the so-named parameter in a Boltzmann distribution. Applied to text, generative models trained in this fashion produce surprisingly coherent passages that appear to reflect the characteristics of the training corpus. They can also be used to continue passages given some starting tokens.

3.2 Concatenated Input Recurrent Neural Networks

Our goal is to generate text in a supervised fashion, conditioned on an auxiliary input $\boldsymbol{x}_{aux}$. This has been done at the word-level with encoder-decoder models (Figure 2b), in which the auxiliary input is encoded and passed as the initial state to a decoder, which then must preserve this input signal

across many sequence steps (Sutskever et al., 2014; Karpathy and Fei-Fei, 2014). Such models have successfully produced (short) image captions, but seem impractical for generating full reviews at the character level because signal from $\boldsymbol{x}_{aux}$ must survive for hundreds of sequence steps.

We take inspiration from an analogy to human text generation. Consider that given a topic and told to speak at length, a human might be apt to meander and ramble. But given a subject to stare at, it is far easier to remain focused. The value of re-iterating high-level material is borne out in one study, Surber and Schroeder (2007), which showed that repetitive subject headings in textbooks resulted in faster learning, less rereading and more accurate answers to high-level questions.

Thus we propose a simple architecture in which input $\boldsymbol{x}_{aux}$ is concatenated with the character representation $\boldsymbol{x}_{char}^{(t)}$. Given this new input $\boldsymbol{x'}^{(t)} = [\boldsymbol{x}_{char}^{(t)}; \boldsymbol{x}_{aux}]$ we can train the model precisely as with the standard generative RNN (Figure 2c). At train time, $\boldsymbol{x}_{aux}$ is a feature of the training set. At predict time, we fix some $\boldsymbol{x}_{aux}$, concatenating it with each character sampled from $\hat{\boldsymbol{y}}^{(t)}$. One might reasonably note that this replicated input information is redundant. However, since it is fixed over the course of the review, we see no reason to require the model to transmit this signal across hundreds of time steps. By replicating $\boldsymbol{x}_{aux}$ at each input, we free the model to focus on learning the complex interaction between the auxiliary input and language, rather than memorizing the input.

3.3 Weight Transplantation

Models with even modestly sized auxiliary input representations are considerably harder to train than a typical unsupervised character model. To overcome this problem, we first train a character model to convergence. Then we transplant these weights into a concatenated input model, initializing the extra weights (between the input layer and the first hidden layer) to zero. Zero initialization is not problematic here because symmetry in the hidden layers is already broken. Thus we guarantee that the model will achieve a strictly lower loss than a character model, saving (days of) repeated training. This scheme bears some resemblance to the pre-training common in the computer vision community (Yosinski et al., 2014). Here, instead of new output weights, we train new input weights.

3.4 Running the Model in Reverse

Many common document classification models, like tf-idf logistic regression, maximize the likelihood of the training labels given the text. Given our generative model, we can then produce a predictor by reversing the order of inference, that is by maximizing the likelihood of the text, given a classification. The relationship between these two tasks ($P(\boldsymbol{x}_{aux}|\text{Review})$ and $P(\text{Review}|\boldsymbol{x}_{aux})$) follows from Bayes' rule. That is, our model predicts the conditional probability $P(\text{Review}|\boldsymbol{x}_{aux})$ of an entire review given some $\boldsymbol{x}_{aux}$ (such as a star rating). The normalizing term can be disregarded in determining the most probable rating and when the classes are balanced, as they are in our test cases, the prior also vanishes from the decision rule leaving $P(\boldsymbol{x}_{aux}|\text{Review}) \propto P(\text{Review}|\boldsymbol{x}_{aux})$.

4 Experiments

All experiments are executed with a custom recurrent neural network library written in Python, using Theano (Bergstra et al.) for GPU acceleration. Our networks use 2 hidden layers with 1024 nodes per layer. During training, examples are processed in mini-batches and we update weights with RMSprop (Tieleman and Hinton, 2012). To assemble batches, we concatenate all reviews in the training set together, delimiting them with (<STR>) and (<EOS>) tokens. We split this string into mini-batches of size 256 and again split each mini-batch into segments with sequence length 200. Furthermore, LSTM state is preserved across batches during training. To combat exploding gradients, we clip the elements of each gradient at $\pm$ 5. We found that it was faster to first train the concatenated input model if we first trained an unsupervised character-level generative RNN to convergence. We then transplant weights from the unsupervised net to initialize the concatenated-input RNN. We implement two nets in this fashion, one using the star rating scaled to [-1, 1] as $\boldsymbol{x}_{aux}$, and a second using a one-hot encoding of 5 beer categories as $\boldsymbol{x}_{aux}$.

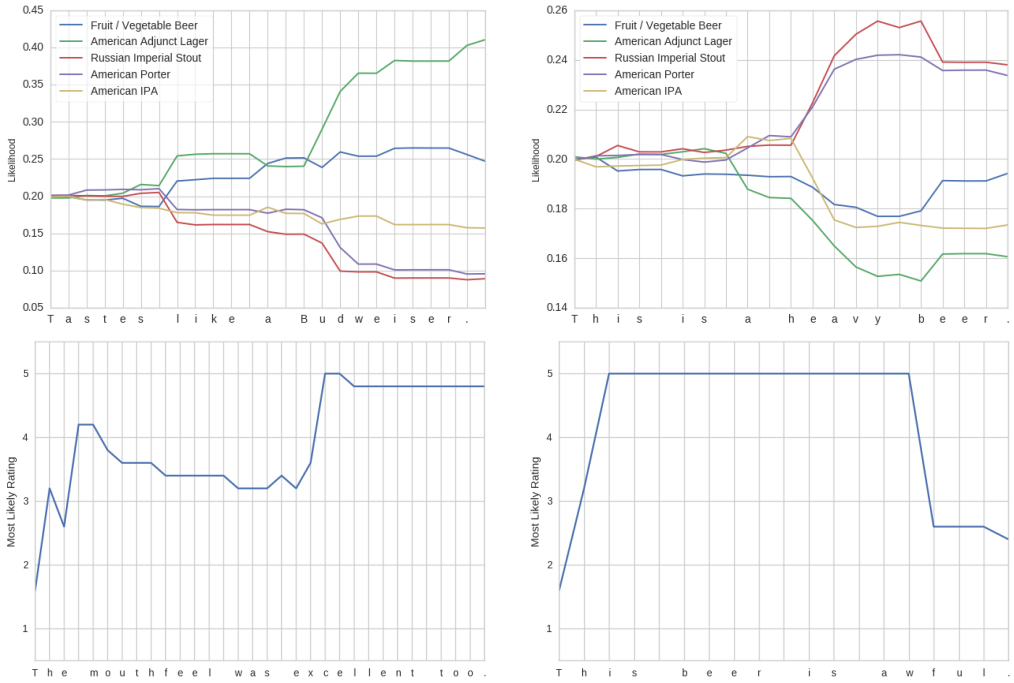


Figure 3: (top) Probability of each category and (bottom) most likely star rating as each letter is encountered. The RNN learns Budweiser is a lager and that stouts and porters are heavy. It learns to tilt positive by the 'c' in 'excellent' and that the 'f' in 'awful' reveals negative sentiment.

4.1 Generating Text

Running the concatenated input RNN in generative mode and conditioning upon a 5 star rating, we produce a decidedly positive review:

> <STR>Poured from a 12oz bottle into a pint glass. A: Pours a deep brown color with a thin tan head. The aroma is of coffee, chocolate, and coffee. The taste is of roasted malts, coffee, chocolate, and coffee. The finish is slightly sweet and smooth with a light bitterness and a light bitterness that lingers on the palate. The finish is slightly bitter and dry. Mouthfeel is medium bodied with a good amount of carbonation. The alcohol is well hidden. Drinkability is good. I could drink this all day long. I would love to try this one again and again. <EOS>

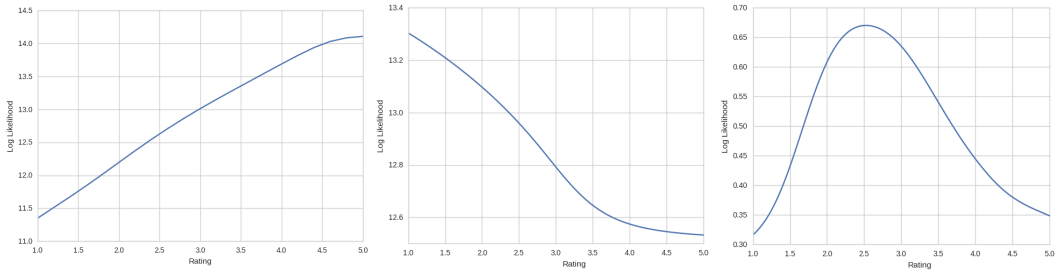
Conditioning on the "Fruit / Vegetable Beer" category, the model generates a commensurately botanical review; interestingly the user "Mikeygrootia" does not exist in the dataset.

> <STR>Thanks to Mikeygrootia for the opportunity to try this one. A: Poured a nice deep copper with a one finger head that disappears quickly. Some lacing. S: A very strong smelling beer. Some corn and grain, some apple and lemon peel. Taste: A very sweet berry flavor with a little bit of a spice to it. I am not sure what to expect from this beer. This stuff is a good summer beer. I could drink this all day long. Not a bad one for me to recommend this beer.<EOS>

For more examples of generated text, please see Appendix A and Appendix B.

4.2 Predicting Sentiment and Category One Character at a Time

In addition to running the model to generate output, we take example sentences from unseen reviews and plot the rating which gives the sentence maximum likelihood as each character is encountered (Figure 3). We can also plot the network's perception of item category, using each category's prior and the review's likelihood to infer posterior probabilities after reading each character. These visualizations demonstrate that by the "d" in "Budweiser", our model recognizes a "lager". Similarly,



(a) "Mindblowing experience." (b) "Tastes watered down." (c) "Not the best, not worst."

Figure 4: Log likelihood of the review for many settings of the rating. This tends to be smooth and monotonic for unambiguous sentences. When the sentiment is less extreme, the peak is centered.

reading the "f" in "awful", the network seems to comprehend that the beer is "awful" and not "awesome" (Figure 3). See appendices C and D for more examples.

To verify that the argmax over many settings of the rating is reasonable, we plot the log likelihood after the final character is processed, given by a range of fine-grained values for the rating (1.0, 1.1, etc.). These plots show that the log likelihood tends to be smooth and monotonic for sentences with unambiguous sentiment, e.g., "Mindblowing experience", while, they are smooth with a peak in the middle when sentiment is ambiguous , e.g., "not the best, not the worst." (Figure 4). We also find that the model understands nonlinear dynamics of negation and can handle simple spelling mistakes, as seen in Appendices E and D.

4.3 Classification Results

While our motivation is to produce a character-level general model, running in reverse-fashion as a classifier proved an effective way to objectively gauge what the model *knows*. To investigate this capability more thoroughly, we compared it to a word-level tf-idf n-gram multinomial logistic regression (LR) model, using the top 10,000 n-grams. Our model achieves a classification accuracy of 89.9% while LR achieves 93.4% (Table 1). Both models make the majority of their mistakes confusing Russian Imperial Stouts for American Porters, which is not surprising because a stout is a sub-type of porter. If we collapse these two into one category, the RNN achieves 94.7% accuracy while LR achieves 96.5%. While the reverse model does not yet eclipse a state of the art classifier, it was trained at the character level and was not optimized to minimize classification error or with attention to generalization error. In this light, the results appear to warrant a deeper exploration of this capability. Please see Appendix F for detailed classification results. We also ran the model in reverse to classify results as positive (≥ 4.0 stars) or negative (≤ 2.0 stars), achieving AUC of .88 on a balanced test set with 1000 examples.

		Predicted Label				
		F/V	Lager	Stout	Porter	IPA
True Label	F/V	910	28	7	14	41
	Lager	50	927	3	3	17
	Stout	16	1	801	180	2
	Porter	22	3	111	856	8
	IPA	19	12	4	12	953

Table 1: Confusion matrix for classifying reviews by beer category with the generative model.

5 Related Work

The prospect of capturing meaning in character-level text has long captivated neural network researchers. In the seminal work, "Finding Structure in Time", Elman (1990) speculated, "one can ask whether the notion 'word' (or something which maps on to this concept) could emerge as a consequence of learning the sequential structure of letter sequences that form words and sentences (but in which word boundaries are not marked)." In this work, an 'Elman RNN' was trained with 5 input

nodes, 5 output nodes, and a single hidden layer of 20 nodes, each of which had a corresponding context unit to predict the next character in a sequence. At each step, the network received a binary encoding (not one-hot) of a character and tried to predict the next character's binary encoding. Elman plots the error of the net character by character, showing that it is typically high at the onset of words, but decreasing as it becomes clear what each word is. While these nets do not possess the size or capabilities of large modern LSTM networks trained on GPUs, this work lays the foundation for much of our research. Subsequently, in 2011, Sutskever et al. (2011) introduced the model of text generation on which we build. In that paper, the authors generate text resembling Wikipedia articles and New York Times articles. They sanity check the model by showing that it can perform a *debagging* task in which it unscrambles bag-of-words representations of sentences by determining which unscrambling has the highest likelihood. Also relevant to our work is Zhang and LeCun (2015), which trains a strictly discriminative model of text at the character level using convolutional neural networks (LeCun et al., 1989; 1998). Demonstrating success on both English and Chinese language datasets, their models achieve high accuracy on a number of classification tasks.

Related works generating sequences in a supervised fashion generally follow the pattern of Sutskever et al. (2014), which uses a word-level encoder-decoder RNN to map sequences onto sequences. Their system for machine translation demonstrated that a recurrent neural network can compete with state of the art machine translation systems absent any hard-coded notion of language (beyond that of words). Several papers followed up on this idea, extending it to image captioning by swapping the encoder RNN for a convolutional neural network (Mao et al., 2014; Vinyals et al., 2015; Karpathy and Fei-Fei, 2014).

5.1 Key Differences and Contributions

RNNs have been used previously to generate text at the character level. And they have been used to generate text in a supervised fashion at the word-level. However, to our knowledge, this is the first work to demonstrate that an RNN can generate relevant text at the character level. Further, while Sutskever et al. (2011) demonstrates the use of a character level RNN as a scoring mechanism, to our knowledge, this is the first paper to use such a scoring mechanism to infer labels, simultaneously learning to generate text and to perform supervised tasks like multiclass classification with high accuracy. Our work is not the first to demonstrate a character-level classifier, as Zhang and LeCun (2015) offered such an approach. However, while their model is strictly discriminative, our model's main purpose is to generate text, a capability not present in their approach. Further, while we present a preliminary exploration of ways that our generative model can be used as a classifier, we do not train it directly to minimize classification error or generalization error, rather using the classifier interpretation to validate that the generative model is in fact modeling the auxiliary information meaningfully.

6 Conclusion

In this work, we demonstrate the first *character-level* recurrent neural network to generate relevant text conditioned on auxiliary input. This work is also the first work, to our knowledge, to generate coherent product reviews conditioned upon data such as rating and item category. Our quantitative and qualitative analysis shows that our model can accurately perform sentiment analysis and model item category. While this capability is intriguing, much work remains to investigate if such an approach can be competitive against state of the art word-level classifiers. The model learns nonlinear dynamics of negation, and appears to respond intelligently to a wide vocabulary despite lacking any *a priori* notion of words.

We believe that this is only beginning of this line of research. Next steps include extending our work to the more complex domain of individual items and users. Given users with extensive historical feedback in a review community and a set of frequently reviewed items, we'd like to take a previously unseen (user, item) pair and generate a review that plausibly reflects the user's tastes and writing style as well as the item's attributes. We also imagine an architecture by which our concatenated input network could be paired with a neural network encoder, to leverage the strengths of both the encoder-decoder approach and our approach. Details of this proposed model are included in Appendix G.

7 ACKNOWLEDGEMENTS

Zachary C. Lipton's research is funded by the UCSD Division of Biomedical Informatics, via NIH/NLM training grant T15LM011271. Sharad Vikram's research is supported in part by NSF grant CNS-1446912. We would like to thank Professor Charles Elkan for his mentorship. We gratefully acknowledge the NVIDIA Corporation, whose hardware donation program furnished us with a Tesla K40 GPU, making our research possible.

REFERENCES

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A cpu and gpu math compiler in python.

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

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10):2451–2471, 2000.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

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

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*, 2014.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

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

Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-RNN). *arXiv preprint arXiv:1412.6632*, 2014.

Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, pages 897–908. International World Wide Web Conferences Steering Committee, 2013.

John R. Surber and Mark Schroeder. Effect of Prior Domain Knowledge and Headings on Processing of Informative Text. *Contemporary Educational Psychology*, 32(3):485–498, jul 2007. ISSN 0361476X. doi: 10.1016/j.cedpsych.2006.08.002. URL `http://www.sciencedirect.com/science/article/pii/S0361476X06000348`.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

Tijmen Tieleman and Geoffrey E. Hinton. Lecture 6.5- RMSprop: Divide the gradient by a running average of its recent magnitude. https://www.youtube.com/watch?v=LGA-gRkLEsI, 2012.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

Appendices

A Reviews Generated by Item Category

In the following examples, we generate reviews conditioned on the item category. We show one example for each for American IPA, American Adjunct Lager, Russian Imperial Stout, American Porter, and Fruit/Vegetable Beer. All reviews in this section are generated with a temperature of .5

A.1 American IPA

<STR>Pours a transparent copper with a nice creamy off-white head that retains well and leaves behind a nice lace. S: Mild aroma of citrus and pine hops. The smell is not super pleasing. T: The body isn't overly abused by the orange peel flavors. It is a solid IPA that is followed by a mouthfeel with a huge rich malt background that is quite prevelent. The finish is slightly bitter on the tongue, and the malt is slightly dry by the end. A touch of sweet malts in the background. M: Great carbonation and a light moderate carbonation feel. Flavors are very pleasant nicely balanced by the hops and resins. D: This is an easy drinking IPA and I would definitely like to try it again for what it is anyway. Overall: This is a well crafted beer. It is a great example of an IPA, I would definitely have another one of these and I will be getting another can in the bottle and like to see it more in the can in some time.<EOS>

A.2 American Adjunct Lager

<STR>A - Poured a dark yellow color with a white head that leaves lacing behind S - Lightest aroma, some corn with a slight hop presence. It has a fairly strong aroma of corn and yeast. The taste is slightly sweet and crisp, corn and a light hop presence. The finish is moderately bitter with a slight sweetness. Finishes sweet and clean with a light crisp finish. Not much more to say. This would be a good one to have when ice cold, and I'd rather have a Coors Light.<EOS>

A.3 Russian Imperial Stout

<STR>Thanks to Buttlepoptu for the chance to try this one. Got this bottle from a 12 oz. bottle in a tasting glass. A- Pours a deep black color with a miniscule head that dissipates quickly. No lacing to speak of either. S- Nice bitter chocolate nose. Lots of chocolate, coffee, roasted coffee, and a little chocolate. T- The dark roasted malts provide a sweet finish, and in the most refreshing taste and the bitterness of this one is a little subdued. M: Thick and rich, like syrup. Slick and creamy. Sits right away and it lasts for just a little D: What a world class RIS. While I love all the world of beers that I've tried in the past, this is very drinkable. I think it could be a little much for sure. Still a good beer for sure. This one is worth the wait. While it is not a beer I could drink a few of without getting a little difficult to drink and finish the whole bottle. <EOS>

A.4 American Porter

<STR> Served a beautiful black with a mocha head that started about 1/1 inch thick and left lots of lacing. The smell was sweet and roasty with some vanilla and a hint of chocolate. The taste was primarily that of smoky chocolate and some chocolate and barrel character. It had a light mouthfeel with a crisp carbonation. Overall this was a very good beer. I was more interested in pouring this one from a bottle though it is a very good brew (that a must have again). I have another one on tap and it was my ease of time. Overall: A big thanks to Richmond for the chance to try this one. I've enjoyed other porters to be but this is definitely a porter. <EOS>

A.5 FRUIT/VEGETABLE BEER

<STR>On tap at the brewpub. A nice dark red color with a nice head that left a lot of lace on the glass. Aroma is of raspberries and chocolate. Not much depth to speak of despite consisting of raspberries. The bourbon is pretty subtle as well. I really don't know that I find a flavor this beer tastes like. I would prefer a little more carbonization to come through. It's pretty drinkable, but I wouldn't mind if this beer was available. <EOS>

B REVIEWS GENERATED BY STAR RATING

Below, we show examples of the concatenated input network's ability to generate decidedly positive reviews when conditioned on high star ratings and decidedly negative reviews when conditioned on low ratings. Because the star rating is simply a scalar input to the network, we can also generate reviews conditioned on the more extreme 0 star and 10 star ratings, even though neither truly exist in the dataset. We present examples of reviews generated by the model and conditioned upon 0, 1, 5, and 10 star ratings. For each we show an example each for three settings of the temperature (.1, .5, 1.0).

B.1 0 STAR REVIEWS

The following full reviews are generated conditioned on a 0 star rating for varying settings of the temperature.

With a temperature of $.1$

> <STR>Pours a clear amber with a thin white head that dissipates quickly. Smell is of caramel malts and some hops. Taste is a little sweet and malty. The mouthfeel is thin and watery. This is a very disappointing beer. I would not buy this again.<EOS>

With a temperature of $.5$:

> <STR>This is an old bottle of a premium beer. This was a bad batch. Pours a color that is black. The head is a small bubbling of bubbles with a thick ring and leaves some nice lacing. The aroma is a strong metallic twang and some light chocolate notes. The smell is of cocoa and coffee. While the aroma is an astringent blast of chocolate and coffee on the nose. The taste is also filled with a very strong alcohol flavor. It tastes like a bad barleywine. The flavors are very strong, and the malt and dark fruits take over the flavor. The mouthfeel is thin and watery. The drinkability is not good.<EOS>

With a temperature of 1.0

> <STR>The color is a rice golden the body with fizzy yellow carbonation (which is unpleasant). No lacing, which suffers from half an inch. Anyway: horrible, suck, surprise, and unpleasant. I below a dubbel, but the honey seems a bit distracting. Thin almost taste of baked bread, caramel malt and almonds...very sickly without much addition into the bottom of perhaps in my mouth. This beer conversely, but then again my rear sourness really kicked up the malt and lies. Mouthfeel was fine. I was difficult to finish this fixe a year and thank Guinness at a party beer. I say this is one very watered down beer.<EOS>

B.2 1 STAR REVIEWS

With a temperature of 0.1

> <STR>Pours a clear amber with a thin white head. Smell is of sweet malt, and a hint of caramel. Taste is also sweet and malty with a slight hop bitterness and a slight bitterness. Mouthfeel is medium bodied with a smooth and creamy mouthfeel. This is a beer that I will probably not go out of my way for any time soon.<EOS>

With a temperature of 0.5

> <STR>A: Pale yellow color with a small white head. S: Spicy and citrusy. T: Sweet and sour fruit with a bit of a metallic taste at the end. M: Light, sticky, and syrupy. D: Somewhat tasty, but the alcohol does not have a lot of complexity to the beer. I would not buy this again.<EOS>

With a temperature of 1.0

> <STR>This beer was gone alright. It did not go away. It's all about the booze, but it seems a rather peppery, aggressive, heavy, smelling (at times) for something better. No where near as big as I'd like. Artificially. You would love this beer... even a sipper. If kolsh were an oddball sourness, I'd swear this one would win. I haven't been afraid of my fridge selection for less money.<EOS>

B.3 5 STAR REVIEWS

With a temperature of 0.1

> <STR>Poured from a 12 oz. bottle into a pint glass. Appearance: Pours a clear golden color with a thick frothy white head that slowly fades to lacing. The aroma is great. It has a sweet caramel malt aroma with a slight hop aroma. The taste is a little sweet with a slight hop bitterness that lingers in the aftertaste. The mouthfeel is medium bodied with a good amount of carbonation. This is a very drinkable beer. I would have no problem drinking a few of these in a sitting. I would definitely drink this again in the future.<EOS>

With a temperature of 0.5

> <STR>A - Poured a midnight black with a thick frothy tan head that left lacing on the glass. S- Smell is a wonderful virtual malt aroma with a touch of coffee and chocolate. Taste is chocolate and coffee with a hint of chocolate and coffee. Its a little thin for a witbier at all. M- This beer has a medium mouthfeel with a slight amount of carbonation. D - Very drinkable beer. I could drink this all night long. I will definitely be buying a few more in the cellar.<EOS>

With a temperature of 1.0

> <STR>First off, I have batch whose pure vertical esopetime trade! The forgot "Homaching copper" IPA would be a popular beer fixed in Pounds VE! Poured into Duvel tulip. APV of caramel with a small light tan head that cakes off on the glass now and lays down to a thin layer of residence that leaves behind dots of lace as it settles. Several left behind with remarkable introducing curtains. Aroma is enveloping massive but incredibly were redolent – refreshing. Juicy Belgian yeast esters - bursts of lime, lemon pith, etc... all right up front. Spiciness is in a glorious essence. Really man's Belgian pale and wild yeast funk too. Basically the taste's aroma is mostly grapefruit juice and vegetably accents. hints of oak, cranberries and light hints of yeast funk. A faint funkiness in there as well, with all malts. And smooth and a bit muddy too. Finish is fresh and both easy to glass, you know the glass who dies on that sweetness itself, but you still think you what effect you get - it makes the least busy-love color? Stone has perfect! Mouthfeel is smooth and clean with an earthy smoothness that lack the nice sharp bite of the malt and certain to exquisitely surface from the malt. Exceptional imperial stout. Low to drinkability, in an awesome winter warmer, especially when a bit of age doesn't work for me. A 20% beer is only great by my someone worry but a beer that would go well with something along with my friend. Nicely done in any mark. Superb.<EOS>

B.4 10 STAR REVIEWS

With a temperature of 0.1

> <STR>Poured from a 12 oz. bottle into a pint glass. Appearance: Pours a dark brown color with a thick tan head that lasts forever and leaves a nice lacing on the glass. The aroma is of chocolate, coffee, chocolate, and a hint of chocolate. The taste is a nice blend of coffee, chocolate, and coffee. The mouthfeel is smooth and creamy with a nice creamy feel to it. This is a very good beer that I would

> definitely have again. I would love to have this again and again. I wish I had more of this stuff and see how this is the best beer I've ever had. I could drink this all day and enjoy it all the time. I could drink this all day and drink all day. I could drink this all day and drink all day and savor the other three of these and this is a staple in my fridge.<EOS>

With a temperature of 0.5

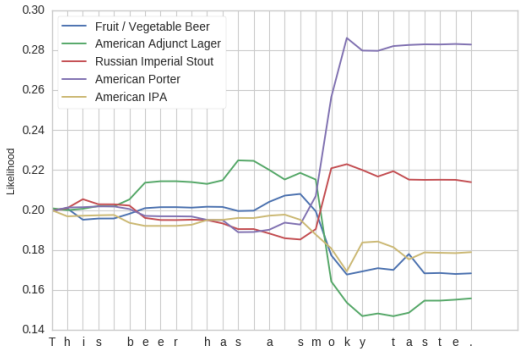
> <STR>A- Amber brown with a small white head that has decent retention and leaves a nice lacing on the glass. Aroma is strong of citrus hops and a mild sweet malt aroma. The taste is so smooth and crisp tasting with the hops throughout the taste. The malt is more citrus than I expect to be sure this is a hop bomb with the alcohol in this beer that makes it easy to drink and smooth. The alcohol is noticeable and delicious. This is a great beer, and a great take on the style and one of the best in the world. I could drink this all night. I guess the alcohol is evident in the flavor and the flavors are bright and crisp and drinks smoother and is so easy to drink. I wish I had more! This is the best beer that I've ever had in awhile. I would definitely order it again and again.<EOS>

With a temperature of 1.0

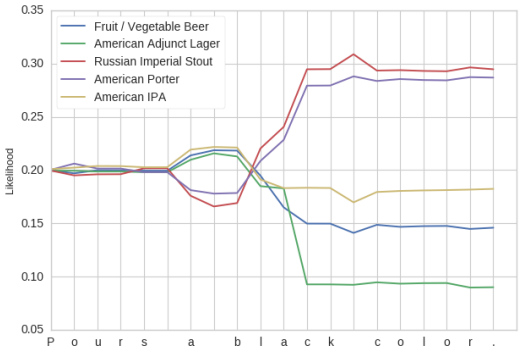
> <STR>A: Pours vibrant amber with a nice, hugs foamy head. Awesome light brown sheen that looks like it. successful excellent! Want down thoroughly IPA indeed. Soft with pine and lime juice (since you sat down!) come through at most. Also kicking bach down from something dampeness. Take through the enjoyment of this is wonderful for sure. Delicious. Is this rank? Now that it full body's imperessively. Espresso-strock, this beer is that fantastic. You will nip sip that OK. I drink one beer yet! I can't remember the 2008 version I might even paybe local!<EOS>

C CATEGORY TRACKING OVER SENTENCES

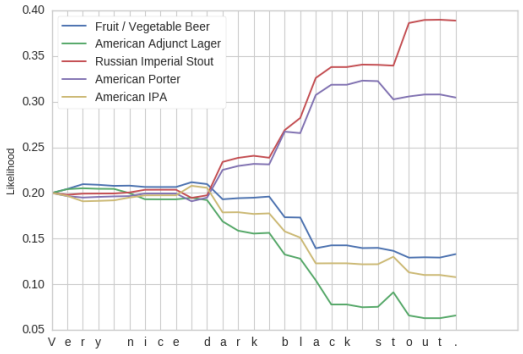
Here we show several representative examples demonstrating the ability of the concatenated input network (run in reverse) to recognize the category of the beer at the precise point in the sentence when it becomes clear. At first the probabilities are all close to .2 reflecting the uniform prior. By the end of the sentences the distribution conditioned on the input is considerably less entropic.



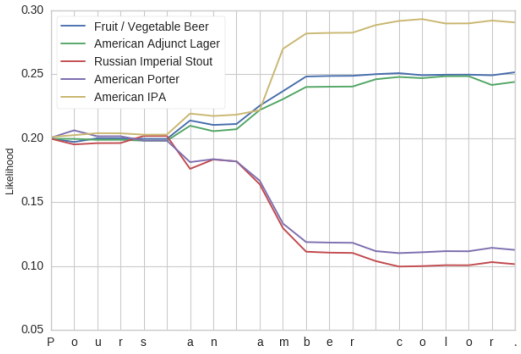
(a) "This beer has a smoky taste."



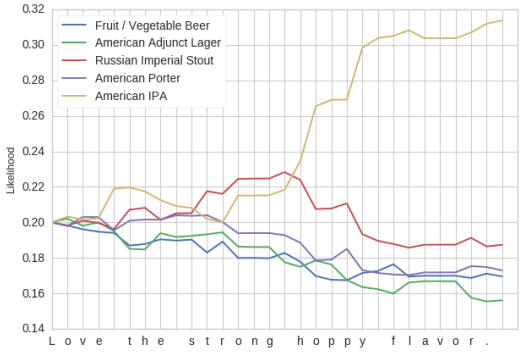
(b) "Pours a black color."



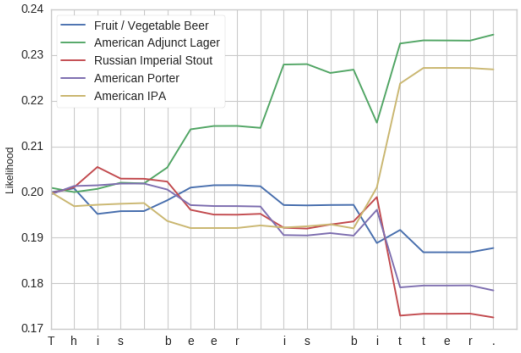
(c) "Very nice dark black stout."



(d) "Pours an amber color."



(e) "Love the strong hoppy flavor."

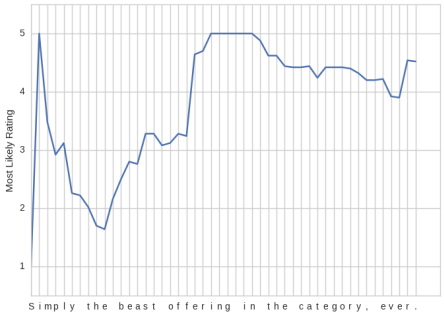


(f) "This beer is bitter."

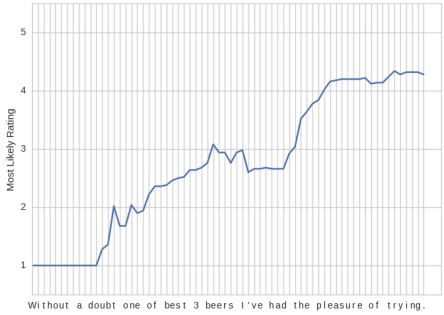
Figure 5: We plot the probabilities of the 5 beer categories after each character in the sentence is encountered.

D SENTIMENT TRACKING OVER SENTENCES

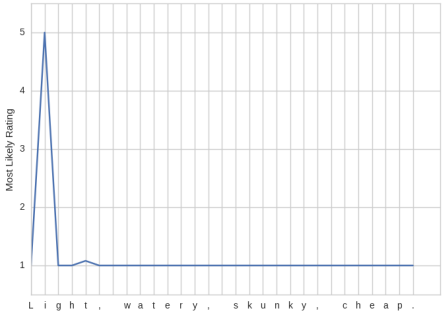
As each character in a review is encountered, we can plot the rating (with a granularity of 100 evenly spaced settings between 1 star and 5) which gives the review highest likelihood. Thus we can tell not only the sentiment of the rating, but the precise word, and even character at which this sentiment became clear.



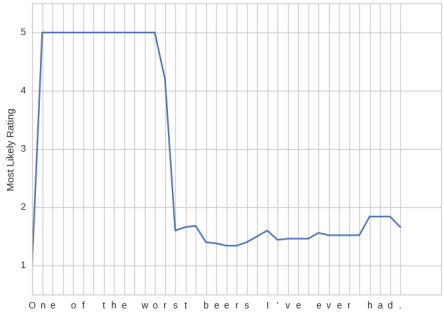
(a) "Simply the beast offering in the category ever."



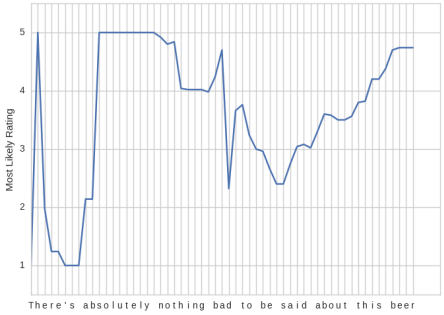
(b) "Without a doubt one of the best 3 beers I've had the pleasure of trying."



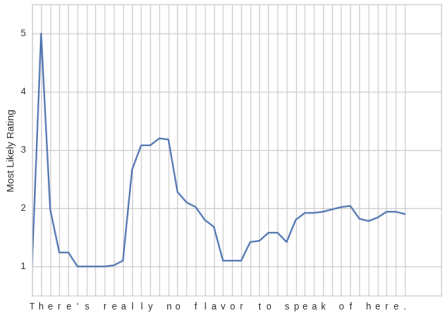
(c) "Light, watery, skunky, cheap."



(d) "One of the worst beers I've ever had.'



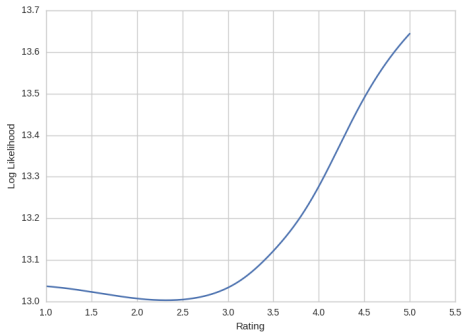
(e) "There's absolutely nothing bad to be said about this beer."



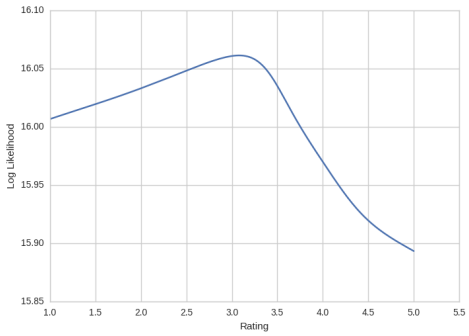
(f) "There's really no flavor to speak of here.'

Figure 6: We plot the argmax of the review's likelihood over many settings of the rating.

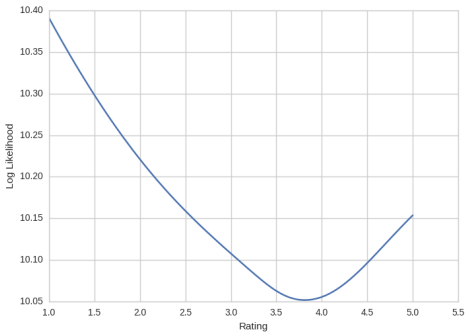
E NONLINEAR DYNAMICS OF NEGATION



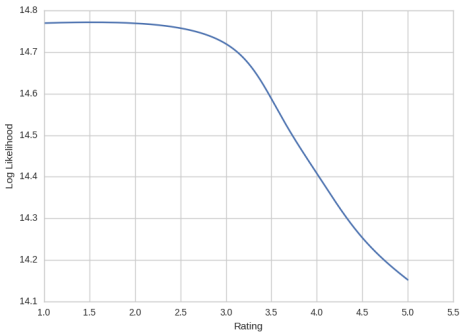
(a) "This beer is great."



(b) "This beer is not great"



(c) "This beer is bad."



(d) "This beer is not bad."

Figure 7: We plot the likelihood given to a review by each rating. The network learns nonlinear dynamics of negation. "Not" reduces the rating when applied to "great" but increases the rating when applied to "bad".

F CLASSIFICATION RESULTS

We trained a concatenated input RNN, with item category information as the auxiliary input. Inferring the class probability via the conditional likelihoods of the review, we can use the model *in reverse* to predict the category of the beer described in the review. Using a balanced test set of 5000 reviews, we evaluated the classification performance of the category RNN against two multinomial regression classifiers, one trained on the top 10,000 n-grams from the training set, and the other trained on tf-idf transfromed n-grams. The confusion matrices for these experiments can be seen in Table 2, Table 3, and Table 4. We also show results for a concatenated input RNN with rating information as used to classify positive (≥ 4.0 stars) and negative (≤ 2.0 stars) reviews.

		Predicted Label				
		F/V	Lager	Stout	Porter	IPA
True Label	F/V	910	28	7	14	41
	Lager	50	927	3	3	17
	Stout	16	1	801	180	2
	Porter	22	3	111	856	8
	IPA	19	12	4	12	953

Table 2: Confusion matrix when classifying item category using the generative model in reverse.

		Predicted Label				
		F/V	Lager	Stout	Porter	IPA
True Label	F/V	916	40	6	15	23
	Lager	29	961	1	1	8
	Stout	11	3	884	100	2
	Porter	16	6	104	870	4
	IPA	20	17	3	5	955

Table 3: Confusion matrix for item category classification with n-gram model.

		Predicted Label				
		F/V	Lager	Stout	Porter	IPA
True Label	F/V	923	36	9	10	22
	Lager	16	976	0	1	7
	Stout	9	4	920	65	2
	Porter	11	6	90	887	6
	IPA	18	13	1	2	966

Table 4: Confusion matrix for item category classification using n-gram tf-idf model.

		Predicted Label	
		Negative	Positive
True Label	Negative	294	206
	Positive	7	493

Table 5: Positive(≥ 4 stars)/ negative (≤ 2 stars.) classification results for RNN

		Predicted Label	
		Negative	Positive
True Label	Negative	464	36
	Positive	191	309

Table 6: n-gram tf-idf positive/negative classification results trained without balancing dataset.

		Predicted Label	
		Negative	Positive
True Label	Negative	459	41
	Positive	42	458

Table 7: n-gram tf-idf positive/negative classification results on balanced dataset.

G A PROSPECTIVE ENCODER CONCATENATION NETWORK

In this paper, we introduced and demonstrated the efficacy of a simple technique for incorporating auxiliary information $\boldsymbol{x_{aux}}$ in a generative RNN by concatenating it with with the character representation $\boldsymbol{x}_{char}^{(t)}$ at each sequence step. However, sometimes we don't simply want to generate given a representation $\boldsymbol{x_{aux}}$, but to learn a representation of $\boldsymbol{x_{aux}}$. For example, to generate a character-level caption given an image, we might want to jointly learn a convolutional neural network to encode the image, and a generative RNN to output a caption.

To accomplish this task at the character level, we propose the following network architecture and hypothesize that is will provide the benefits of learning an encoding while preserving our ability to generate long passages at the character level. At train time the $\boldsymbol{x_{aux}}$ is fed to an encoder, whose output is then passed as auxiliary information to a concatenated input network. At prediction time, for any input, the encoding is calculated once, after which the inference problem is identical to that of our demonstrated concatenated input network.

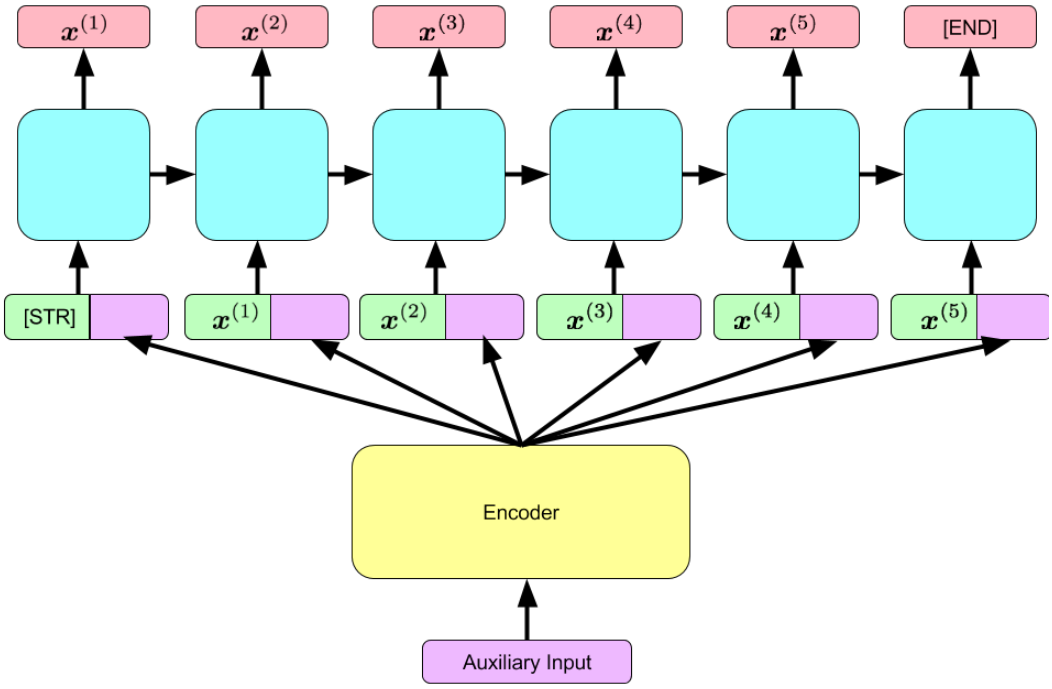
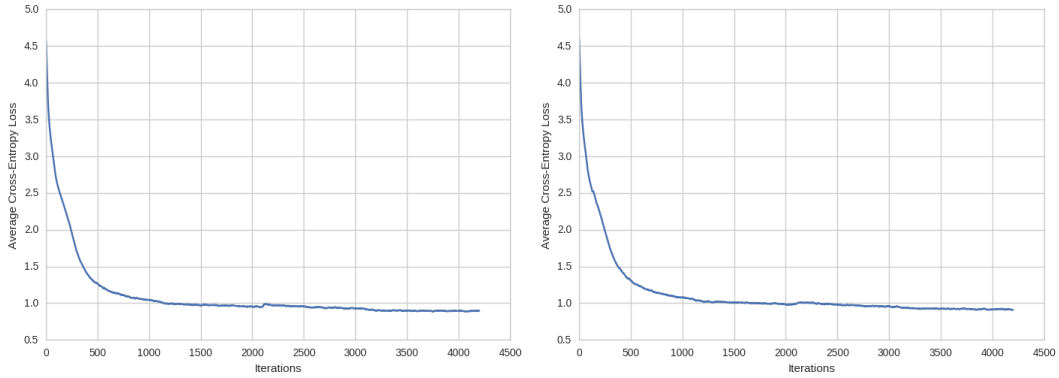


Figure 8: Generative model with input replication. We train the network to produce a *5 star* review by concatenating the rating with the one-hot representation of each character.

H LEARNING CURVES

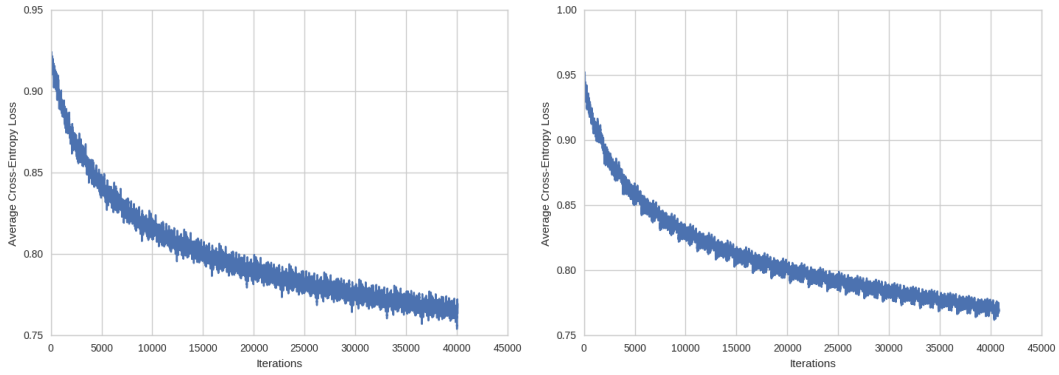
For each task, we train two unsupervised character-level *donor* RNNs so that we may harvest the weights for transplantation into the concatenated input networks. We train separate *donor* networks for the two tasks (rating and category modeling) because each is trained on a different subset of the data (the beer set is selected for class balance among the 5 categories and is thus smaller). These networks are trained until convergence (Figure 9). After transplantation, we train the concatenated input RNNs with high learning rates, to induce the weights for auxiliary information to grow quickly. This results in an initial spike in loss (to quick to be seen in Figure 10), after which the loss quickly decreases.



(a) Unsupervised rating network

(b) Unsupervised category network

Figure 9: The learning curves for the unsupervised donor character RNNs used for weight transplantation.



(a) Rating network after transplantation

(b) Category network after transplantation

Figure 10: The learning curves for rating and category networks after weight transplantation.