

# Transfer Learning and Sentiment Analysis

**Victor Cheung**

Computer Science Department

Stanford University

hoche@stanford.edu

## Abstract

We discuss and evaluate the use of transfer learning in a specific natural language task: sentiment analysis. We use a large and well-trained character-level language model using the multiplicative LSTM architecture as the starting point for the classification task. We find that the hidden representation produced (the last cell state for the last byte in the sequence) is a dense representation suitable for training a linear classifier that results in performance that rivals or exceeds a battery of strong count-based baselines. Thus, we demonstrate support for the hypothesis that language modeling may be conducive to transfer learning across different natural language tasks.

## 1 Introduction

In adjacent and complementary fields to NLP and NLU, such as Computer Vision, neural network based models have come to dominate the frontier of research and usually reach the top of the rankings in metrics used to assess different models. A similar situation has arisen in NLU, where recurrent neural networks (RNN) (along with complex variations on RNN, often mixed with convolutions, highway networks adopted from the CV community, etc.) have proven to be powerful models for sequence modeling. The theoretical justifications for the use of RNNs are legion: they capture long range ordering information (ex. modifiers separated by parenthetical information); allow for variable length inputs and a corresponding number of computations; and learn a dense representation suitable for a variety of tasks (many-to-many as in machine translation, or one-to-many as in image labeling, or many-to-one as

in sentiment analysis, etc). However, widespread adoption and use of RNNs is limited by practical issues, such as difficulties encountered during training due to vanishing or exploding gradients, large parameter counts and concomitant demand on resources, and so on. One way to circumvent these issues is the use of transfer learning.

Transfer learning for neural networks has featured prominently in the computer vision community and is key to its usage in industry. It's been found that weights in the first few layers of the network capture general information that are not task-specific. However, it's not immediately clear what the parallel in the NLP world is. A close analogy might be the general information captured in distributed representations such as GloVe and Word2Vec in different dimensions as the default starting point for most NLP tasks; however, these reflect semantic relationships between words only and are not equivalent to the information captured in the weights of a neural network. Moreover, the concept of transfer learning is still not well-defined and different authors across different domains (Reinforcement Learning in particular) address the same or very similar concepts with different names. For example, the RL community investigates the problem of meta-learning (or learning to learn) which is concerned with building an agent that quickly trains and performs well across different environments. This is similar to what we aim to do, i.e. build a model that is able to quickly train and perform well across reviews from different sources.

In order to assess the use of transfer learning in NLU, we choose the task of sentiment analysis across a variety of domains of review types. Sentiment Analysis is a rapidly growing research area in NLU, and seeks to characterize the nuance and polarity of opinions. Beyond a clear path towards applicability in industry settings (automated

monitoring and assessment of public opinions of politicians, political parties, businesses, brands, celebrities, etc. immediately come to mind), we would suggest that any system truly capable of natural language understanding be able to grasp the whimsy and nuances in human sentiments. We believe that solving sentiment analysis is a key stepping stone to a system capable of understanding text in general.

To build our own sentiment classifier, we will adapt a pretrained RNN model from the literature, featurize our text via a forward pass through the network (treating the review as a sequence of characters), take the final hidden cell representation and train it against a linear model to produce the final label (sentiment). We will do this in both a binarized setting and a full-label setting across a variety of datasets from Amazon, Yelp, and RateBeer. Furthermore, we will seek to answer another question about transfer learning: can the pretrained networks—specifically trained on one domain of reviews (ex. Amazon)—generalize to perform on other types of reviews (ex. beers, restaurant ratings)? That is, can our models learn to differentiate between positive and negative sentiment in a context-invariant way?

## 2 Previous Approaches

Since we will rely heavily on neural networks, we anticipate that training difficulties will be non-trivial, as is often the case. Training difficulties are notorious for RNNs and obviated their use in natural language tasks for some time. Thus, we refer to “Semi-supervised Sequence learning” by (Dai and Le, 2015) sought to overcome unstable training of recurrent neural networks (RNN) on specific tasks. Their solution was to pretrain RNNs using unlabeled data. In fact, Dai et al, were one of the first applications of the recurrent neural networks and autoencoders to pretrain a network that can be later tailored for downstream applications. Their hypothesis was that pretraining a neural network in this way will 1) bypass difficulties usually encountered in actual usage of RNNs for NLP due to instability of training and 2) allow for better performance than RNNs initialized randomly for specific tasks (warm-starting the training process).

Other works have explored similar hypotheses to the one presented above and found that pretraining (either with autoencoders or language models) have resulted in better performance in ma-

chine translation and other tasks. We discussed “Unsupervised Pretraining for Sequence to Sequence Learning” by (Ramachandran et al., 2016) in the literature review, where they present an unsupervised learning method that generally improves the performance of sequence to sequence (seq2seq) tasks. While the focus is primarily on machine translation, the lessons remain relevant for single label classification tasks. Their hypothesis was that initializing an autoencoder with pretrained weights from language models and then fine-tuning by training on a labeled corpus will result in better performance compared to initializing randomly.

However, given the large vocabulary of the English language, we must confront the resulting large parameter counts of the neural network models and slowness of training. “Character-Aware Neural Language Models” by (Kim et al., 2015) addresses directly this problem of large parameter counts of LSTM language models. These models tend to be word-based, and since each word is usually represented by hundreds of dimensions, this results in a large number of parameters for the first LSTM layer. Furthermore, word-based embedding such as GloVe and Word2Vec do not take into account morphological linguistic features. The examples given by the author is “eventful”, “eventfully”, “uneventful”, “uneventfully”. Thus, in a sense, morphological information does not always transfer between word vectors. This character-based language model, by looking at convolutions of various widths, would then capture morphology. Thus, the authors hypothesize that a character-level approach would reveal the connections between these words, as well as significantly reducing the parameter count for the first LSTM layer, since each character can be represented with significantly fewer parameters.

So we see that pretraining has empirically shown to improve performance; however, there remains the problem that pretraining has historically been done on small datasets since neural networks train slowly on large datasets. “Exploring the Limits of Language Modeling” by (Jozefowicz et al., 2016) addresses this problem of training language models on a large scale. Previous work on language modeling focused on the Penn Tree Bank. Its small size made overfitting a difficult problem to avoid. Jozefowicz et al. chose to use the One Billion Word Benchmark, which is much

larger than the Penn Tree Bank in both the corpora (1B words) and the vocabulary size (800k words), as well as significantly more complex and longer structures. While the One Billion Word Benchmark is only considered a medium-sized dataset for count-based LMs, for neural-based LMs this is quite large and presents a challenge. In order to facilitate training on a significantly larger dataset, the primary contribution of Jozefowicz was in designing a new softmax loss based on importance sampling that allowed them to efficiently train a model equivalent in performance to the standard LM model but with significantly fewer parameters. The softmax output of an LM is over the vocabulary, which when very large, as in this case, implies that the computation of the necessary probabilities becomes prohibitively expensive during training. The authors show that with importance sampling we can approximate the partition function (the denominator in softmax) with significantly less computation to make evaluation of the last layer computationally feasible.

To make the pipeline of leveraging pretrained language models in task-specific ways concrete, “Fine-tuned Language Models for Text Classification” by (Howard and Ruder, 2018) propose a specific training regime that outperforms existing state-of-the-art results in sentiment analysis, question classification and topic classification. The techniques proposed are Fine-tuned Language Models (FitLam), Discriminative FineTuning and Back-Propagation Through Time for Text Classification. The idea of FitLAM is to pretrain a large LSTM (for example as proposed by (Jozefowicz et al., 2016)) over a large dataset, then fine-tune the LSTM by training again over the target task distribution. This is done via gradual unfreezing: the last layer is fine-tuned first, then the last two layers are fine-tuned, then the last three layers, and so on until all layers have been retrained. For the task classifier, the authors propose concatenating the max-pooled and mean-pooled representations of the word-level hidden states to the final hidden state prior to the last fully connected layers. As well, the authors note that different layers ought to be trained with different learning rates since each encodes different information. They call this Discriminative Fine-Tuning. For large documents, Howard et al. divided the document into fixed size batches. At the beginning of each batch, they initialize the states with the state at the end of the

last batch. Gradients are backpropagated only to the batches that contributed to the final prediction. They called this Back-Propagation Through Time for Text Classification.

### 3 Data, Model and Experimentation Infrastructure

In order to assess the generality of the pretrained language model, we will assess its performance when asked to do sentiment analysis across a variety of domains of reviews. A Priori, we expect that the model will perform better on the domains it was trained on (Amazon) versus other domains (Yelp, RateBeer). We will uniformly randomly subsample 10,000 reviews from each dataset. We note that since the reviews themselves are not class-balanced, our subsample resembles the original distribution of reviews and is heavily weighted towards higher ratings.

Our experimentation framework is twofold:

1. Label predictions in a binary setting: we assign all ratings below a threshold as 0 (negative), and all ratings above the threshold as 1 (positive). While arbitrary, this method allows us to forcibly resolve some ambiguity in “neutral” reviews.
2. Label predictions in an identity setting: we predict labels that accommodate the range of labels from the original dataset. In particular, we note that the RateBeer dataset has 20 labels (1/20 - 2/20). This setting assists in assessing whether our language model has been able to capture nuances in language in the final hidden representation such that it can distinguish between gradients of sentiments beyond the binary setting.

To accommodate experimentation and our different data sources, we have built infrastructure that allows for easy, modular experimentation much in the same vein as the `sst` framework presented by Chris Potts. We wrote a variety of helper function that facilitate data sources in a variety of data formats. This was critical since we aim to experiment across different review types, which often necessitates taking data from different sources, all stored in different formats. Our utilities allow us to convert these different datasets into a common format suitable for either sklearn model fitting or for training recurrent neural networks. We

leverage sklearn’s many built-in utilities in order to do dataset splitting and cross-validation. We also use sklearn’s built-in text featurizers in order to create sparse feature representations that easily accommodate a number of features: we cap the number of the features (in order to upper bound training time and avoid overfitting, and compute unigrams. We wrote an experimental framework that allows us to take in arbitrary model specifications, feature representations, development and test assessment, and arbitrary scoring functions.

We have also chosen to use the Pytorch framework for building our neural network due to its optimized performance relative to Tensorflow. We have found pretrained language models ported to Pytorch from previous work and built infrastructure to accommodate loading and featurizing using these models. We find that this is necessary due to resource and time constraints.

### 3.1 Data

Since we use a character-level (byte-level) RNN, there is no need to preprocess the dataset and deal with out of vocabulary words, misspellings, capitalization, and so on. For all datasets, we divide the dataset into a 81% training set, a 9% development set, and a 10% test set. All hyperparameters for the baseline classifiers and the mLSTM-based classifiers are selected over the development set. We do not run on the test set for all models until after choosing hyperparameters for all models.

#### 3.1.1 Amazon

The Amazon dataset is adapted from (He and McAuley, 2016), the same dataset as the corpus used for the language model. We use the deduplicated version (with reviews for near-identical products removed or merged). This dataset only includes reviews for products with more than five reviews and where the users have reviewed more than 5 products (referred to as 5-core). Since the dataset is separated into more than 24 categories, we focus on a selected subset given our time frame for experimentation. Datasets selected include “Instant Video”, “Musical Instruments”, and “Digital Music”. The reviews are quite complex and can range up to hundreds of words and multiple paragraphs.

#### 3.1.2 RateBeer

The reviews from (McAuley et al., 2012) span a period of 10 years and include about 3 million

reviews up to November 2011. User IDs, Beer IDs are given, along with five subjective aspects of the beer including appearance, aroma, palate, taste, all rated out of five. The overall score is a summation of the previous characteristics. The reviews include a plain text description of the product given by the user, and can range up to multiple paragraphs. Users often revised and update their reviews based on extended impressions at a later date. Given the scoring system, the labels for this dataset is considerably more nuanced than either Amazon or Yelp, which are simply rated out of five.

#### 3.1.3 Yelp

The yelp reviews from the Yelp Dataset Challenge in 2015 include data from around 5000 restaurants. Metadata associated with the dataset include anonymized business, user and review ids, the dates of the review, star ratings, and usefulness ratings from other users. Distribution of star ratings is skewed towards one-star and five-star ratings, as expected.

### 3.2 Model

Our primary model for experimentation is the Multiplicative Long Short Term Memory Network first proposed by (Krause et al., 2016). This architecture was specifically chosen since it’s suitable for character-level language modeling, which as in (Kim et al., 2015) allows for us to capture subword information even in a large corpora and large vocabulary context. Conveniently, this allows us to sidestep the issue encountered in (Jozefowicz et al., 2016), which specifically designed an importance sampling-based loss function to train a word-based language model over the Wikitext corpora. Our model is derived from (NVIDIA, 2018).

The mLSTM model architecture is described below. Suppose we have a one-hot vector  $x_t$  at time step  $t$ , then we introduce an intermediate state  $m_t$  that modifies each of the gating units in the LSTM in the following way:

$$m_t = (W_{mx}x_t) \odot (W_{mh}h_{t-1}) \quad (1)$$

$$\hat{h}_t = W_{hx}x_t + W_{hm}m_t \quad (2)$$

$$i_t = \sigma(W_{ix}x_t + W_{im}m_t) \quad (3)$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_t) \quad (4)$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_t) \quad (5)$$

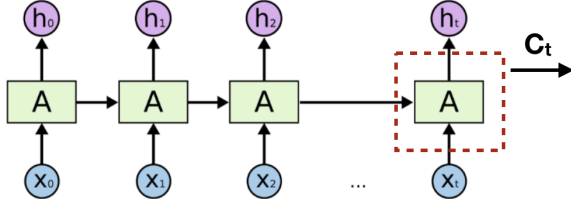


Figure 1: Simplified representation of the unrolled mLSTM network. The internal state  $C_t$  at the last time step is taken as the feature representation of the full sequence. Graphics adapted from (Olah, 2015).

where  $i_t$  is the input gate,  $o_t$  is the output gate,  $f_t$  is the forget gate, and  $\sigma$  is the logistic sigmoid function, all at time step  $t$ . Then we have the usual LSTM structure where

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(\hat{h}_t) \quad (6)$$

$$h_t = \tanh(c_t) \odot o_t \quad (7)$$

thus we have that  $i_t$  determines the information flow from the previous hidden state and the current input to write to the current internal state  $c_t$ ,  $o_t$  determines the amount of information to output to the current hidden state  $h_t$ ,  $f_t$  determines the amount of information to forget from the previous internal state  $c_{t-1}$ . The introduction of  $m_t$  introduces an additional layer of input dependency (via the term  $W_{mx}x_t$  in  $m_t$ ) in the hidden-to-hidden transitions. This implies that it is able to learn parameters contingent on discrete, exclusive inputs at each given time step.

Our model is adapted from a pretrained model trained as a language model on 86 million Amazon reviews. Key characteristics of the pretrained model include persistence of the internal state across folds of size 85829, being reset only after each fold is completed. The hidden states are 4096 dimensions (primarily used in our experiments) and 8192 and the embedding size 64 dimensions. Gradients were clipped at a maximum of 1. With 8 Volta Class GPUs, training took 6.5 for a full epoch across the 86 million reviews.

The model is used to generate a 4096d or 8192d hidden representation of the reviews (processed as a sequence of bytes by the mLSTM). We then train a logistic regression classifier over the hidden representation on the appropriate dataset for the final classifier. This approach is illustrated in [1].

## 4 Results

Our baseline models will be carefully tuned linear models and other decision tree based classifier (Max Entropy (Logistic Regression) classifier, Random Forests, Gradient Boosted Trees, Multinomial Naive Bayes, and SGDClassifier) all with unigram features. These models are called from the SKLearn library. These models were chosen because of their demonstrated performance from previous literature and approaches. Despite their simplicity and speed of training, these well-tuned models rival neural-based approaches in their performance, and with sufficiently rich features, present state-of-the-art results.

From table 1, we see that our neural-based approach is able to exceed our baseline models on three of the datasets; however, we must note that the relative improvement is only marginal. The largest percentage improvement is only 3.2% on Amazon instruments; however, since there is significant random variation in what subsets of the data is chosen to be the training set, the development set and the test set, these results are also subject to variation. Thus, we cannot claim that our approach definitively outperforms the baseline models.

Consider now table 2. Here, we predict over the full range of labels that are available, as opposed to the binary setting. As expected, the overall performance decreases significantly due to the harder task. We find further that recall is consistently better than precision across all the domains. We believe this is due to the imbalance of the different classes, where higher ratings are consistently a larger portion of the training, development and test sets. We suspect that our models have empirically learned this ratio, and thus are more prone to give higher ratings. This implies that more of the higher ratings will be correctly predicted, with the consequence that many of the high ratings predictions are incorrect, leading to lower precision across the board.

Consider now table 3 [4], where we also predict over the full range of labels. We find that the neural-based model also outperforms three of the baseline models on the Amazon Instant, Amazon Instruments and Yelp dataset. However, as in the binarized setting, the improvements are marginal and still subject to random variation in the data. We note the deterioration in performance on the RateBeer dataset, where the percentage change is



Dataset/F1-Score	RF	SGD	SGB	MNB	LOGREG	4096D- mLSTM- LOGREG	PERCENT IMPRO.	(8192D- mLSTM- LOGREG)
AMAZON INSTANT	0.896	0.924	0.904	0.930	<b>0.930</b>	0.952	2.4	
AMAZON MUSIC	0.948	0.958	0.951	<b>0.979</b>	0.957	0.965	(1.4)	
AMAZON INSTRUMENTS	0.925	0.907	0.918	0.928	<b>0.931</b>	0.961	3.2	
YELP	0.830	0.906	0.840	0.893	<b>0.905</b>	0.916	1.2	
RATEBEER	0.875	0.894	0.887	0.904	<b>0.909</b>	0.889	(2.2)	0.888

Table 1: F1-Scores in the binary setting across different datasets. All results are evaluated on the unseen test set. RF is Random Forest, SGD is SGDClassifier, SGB is Gradient Boosted Trees, MNB is Multinomial Naive Bayes, LogReg is Logistic Regression, 4096d-mLSTM-LogReg is the LogReg classifier fitted on featurized text from a 4096d mLSTM model, and similarly for 8192d-mLSTM-LogReg. For the last, we determined that increased number of dimensions are unlikely to result in performance gains following featurization of the RateBeer dataset. Bolded numbers refer to performance of best baseline. Percent improvement is calculated over the best baseline.

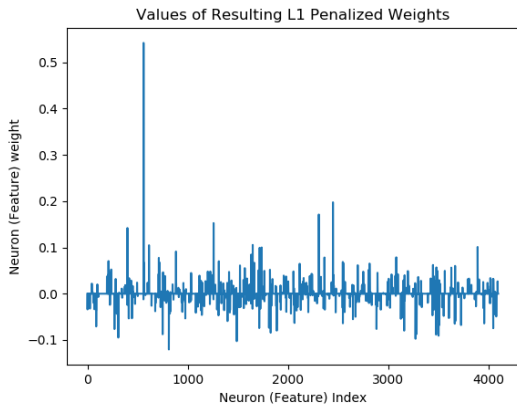


Figure 2: Weight distribution from the L1-penalized Logistic Regression on the hidden representation from the 4096d-mLSTM-LogReg classifier. There is one feature with significantly larger L1 penalty weight, as found in (Radford et al., 2017). This is termed the “sentiment neuron”. See [4].

	PRECISION	RECALL	F1-SCORE
AMAZON INSTANT	0.618	0.623	0.618
AMAZON MUSIC	0.633	0.659	0.642
AMAZON INSTRUMENTS	0.683	0.700	0.690
YELP	0.505	0.510	0.506
RATEBEER*	0.179	0.179	0.175

Table 2: Results from 4096d-mLSTM-LogReg classifier over all available labels. Note that RateBeer is over 20 different classes. All scores are macro-weighted across the size of the support for each label.

due in large part to the difficulty of predicting over 20 possible labels. We note also that the higher dimensional hidden representation actually resulted in worse performance on the RateBeer dataset. We hypothesize that this is due to additional noise introduced by the increased number of dimensions, where a single neuron is sufficient to capture the majority of the sentiment signal (discussed below).

Furthermore, when we examine the L1 penalized weights of the LogReg model in figure 2 [4], we find that there exists one neuron that always has substantially larger penalty than others. This is empirical support of the same phenomenon that arose in (Radford et al., 2017). Empirical studies that shown that L1 penalization promotes sparsity, in the sense of reducing sample complexity — the number of training examples required to learn a specific target function. This is especially true in regimes where there is much noise that do not directly contribute to the classification

Dataset/Model F1-Score	RF	SGD	SGB	MNB	LOGREG	4096D- mLSTM- LOGREG	% IMPROVEMENT	(8192D- mLSTM- LOGREG)
AMAZON INSTANT	0.545	0.549	0.548	0.579	<b>0.595</b>	0.618	3.9	
AMAZON DIGITAL MUSIC	0.559	0.554	0.603	<b>0.658</b>	0.638	0.642	(2.4)	
AMAZON INSTRUMENTS	0.596	0.631	0.616	0.625	<b>0.652</b>	0.690	5.8	
YELP	0.396	0.473	0.460	<b>0.494</b>	0.485	0.506	2.4	
RATEBEER	0.149	0.145	0.177	0.188	<b>0.207</b>	0.175	(15)	0.150

Table 3: F1-Scores in the full labels setting across different datasets. All results are evaluated on the unseen test set. All models are as before. Bolded numbers refer to performance of best baseline. Percent improvement is calculated over the best baseline.

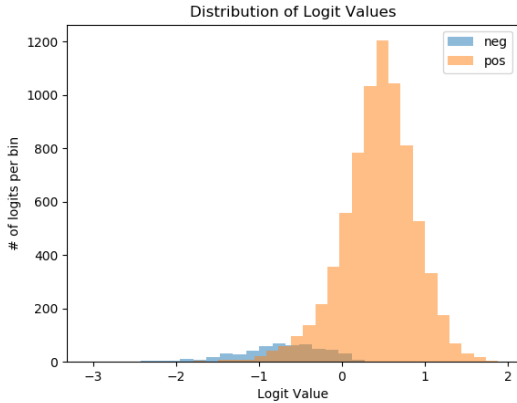


Figure 3: Class distributions contingent on the logit value of the “sentiment neuron”. Below the zero threshold, the “sentiment neuron” is indicative of a negative label; similarly for above the zero threshold, where it is indicative of a positive label. Note the severe class imbalance between positive and negative labels.

task at hand. This is exactly the scenario we are working in: the neural network used was trained for language modeling, not directly for sentiment analysis. Thus, there are likely many signal embedded in the hidden state that would not directly contribute to sentiment analysis. The L1 penalties thus demonstrate the existence of a single feature that strongly indicates either positive or negative sentiment, as in figure 3.

## 5 Discussion

Our results have shown that the hypothesis forwarded in (Dai and Le, 2015) holds. With unsupervised learning via language modeling, we can produce features that result in equivalent performance or exceed the performance of unigram-based linear, SGD and decision-tree based mod-

els. A priori, this is a surprising result. The objective function and the metric of relevance (perplexity) in language modeling are not, en facie, related to the objective function of relevance in sentiment analysis. That the hidden representations (and the signals embedded therein) of the mLSTM model produce a featurization that is conducive to sentiment analysis and other language classification tasks is thus unexpected. However, as suggested by (Jozefowicz et al., 2016), being able to place a distribution over characters (similar to placing a distribution over words), in some sense, captures the complexity of language and also the knowledge within. Thus, language models may in this way “understand” the language being used to express opinions, and produce a dense characterization of it naturally conducive to classification. In our case, as in (Radford et al., 2017), there is evidence that much of the signal regarding sentiment is captured in one neuron, as evident from its activation patterns.

We note further that our neural-based approach consistently performs better on all the Amazon datasets than on either the Yelp dataset or the RateBeer dataset. This supports our initial belief that a language model trained on the same domain as later classification tasks will better suited for that particular distribution than other distributions. The difference in distribution probably arises from selection bias, since the type of users who leave reviews on Amazon products are likely similar. This is substantiated by the performance of our model on the RateBeer dataset. While it still marginal outperforms some of the baseline models (RF, SGD and SGB), it fails to outperform the unigram-based MNB and LogReg models; thus, the hidden representation outputted by our language model on this dataset is likely biased in some way. Furthermore, the characteristics

of the language used to describe beers is different from the language used to describe most Amazon products. Consider some typical reviews from the RateBeer dataset:

- “Pours deep copper with red-ish hue, massive off tan head. Aroma is yeast, hints of banana. Medium moutfeel, alcohol warmth, taste of candy sugar.”
- “Bottled. Clear golden, brief head. Neutral aroma. Medium sweet with some papery malt but near no bitterness. Not very interesting.”

And typical reviews from the Amazon digital music dataset:

- “this has the famous songs from the album back in the day. some of the songs on this CD are i feel the earth move, you’ve got a friend and so far away. and the sound is great. a keeper.”
- “i had this cd back in the days and it broke got me a new one brings back memories. lovely cd”

The evident domain shift implies that our language model may not perform well on the RateBeer domain, since the words used are strongly sensory based. Thus, the resultant hidden representations outputted by the model may be a poor starting point for a linear classifier; nevertheless, we see transfer learning at work where this approach obtains performance that still rivals a subset of the baseline models. We believe this is due to general features of opinion that are captured — i.e., character sequences (words) like “interesting” and “great” and “lovely” general express positive sentiment and are captured by the language model.

## 6 Conclusion

We were interested in understanding and applying transfer learning in the NLU setting. One candidate was the semantic information captured in distributed representations such as GloVe and Word2Vec. These might be the default starting point for most NLP tasks; however, these only reflect semantic relationships between words only and are not equivalent to the information captured in the weights of a neural network, as in the Computer Vision setting. The alternative we explored,

language modeling, proved to be a fruitful method of transfer learning. Despite being trained on the Amazon dataset, it attained good results on domains other than Amazon.

And since the language model we use was first trained on the Amazon dataset, we anticipate that the character embeddings learned in the input layer of the mLSTM is predisposed towards words, phrases and sentences related to opinion expression in the first place. This may explain much of the resultant performance on the Amazon dataset and reviews in other domains. This leads naturally to another question: would a language model trained on a general corpus of the English language, including books, news articles, magazines, etc. be able to produce a hidden representation with similar performance? To assess this, given enough resources, one could conduct a similar experiment with the word-based language model (LSTM based) learned in (Jozefowicz et al., 2016).

Furthermore, one could also repeat our experiments with other character-based recurrent neural networks as in (Kim et al., 2015) that rely on convolutions in the input layer. These enable more explicit modeling of morphemes and subword information, and could result in better performance on the sentiment analysis task depending on the window sizes used in the convolutions. These lead to natural extensions for future work.

## 7 Acknowledgments

We would like to thank Bill MacCartney for being a fascinating lecturer, Christopher Potts for his unending willingness to field questions on Piazza, and the course staff for helping with various aspects of this project.

## 8 Github

Link to code used can be found at [github](#).


## References

- Andrew M. Dai and Quoc V. Le. 2015. [Semi-supervised Sequence Learning](#). *arXiv:1511.01432 [cs]*. ArXiv: 1511.01432.
- Ruining He and Julian McAuley. 2016. [Ups and downs](#). *Proceedings of the 25th International Conference on World Wide Web - WWW '16*.
- Jeremy Howard and Sebastian Ruder. 2018. [Fine-tuned Language Models for Text Classification](#). *arXiv:1801.06146 [cs, stat]*. ArXiv: 1801.06146.



- Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. [Exploring the Limits of Language Modeling](#). *arXiv:1602.02410 [cs]*. ArXiv: 1602.02410.
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. [Character-Aware Neural Language Models](#). *arXiv:1508.06615 [cs, stat]*. ArXiv: 1508.06615.
- Ben Krause, Liang Lu, Iain Murray, and Steve Renals. 2016. [Multiplicative lstm for sequence modelling](#).
- Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1020–1025. IEEE.
- NVIDIA [online]. 2018. [\[link\]](#).
- Christopher Olah. 2015. Understanding lstm networks, 2015. URL <https://colah.github.io/posts/2015-08-Understanding-LSTMs>.
- Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. [Learning to generate reviews and discovering sentiment](#).
- Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. 2016. [Unsupervised Pretraining for Sequence to Sequence Learning](#).

## RateBeer



**MrTipple**  
 8715 Reviews  
 3.9

2 minutes ago

13Jun2018 Meat, leather, sweet oak, ashy, charcoal, dark toasted malts, burnt toast bread, dark roast, burnt coffee. Hazy, very dark brown, tiny, creamy, beige head. Light bitter. More dark, toasted malts, meat, leather, burnt toast, charcoal, earthy, ashy, sweet, dark roast coffee, soft carbonation, full bodied. Very nice, rich, punchy and complex. Bottle shared at L6 tasting - sourced from The Bottle Shop, Bermondsey!

Aroma	<div><div></div><div></div><div></div><div></div><div></div></div>	7/10
Appearance	<div><div></div><div></div><div></div><div></div><div></div></div>	3/5
Taste	<div><div></div><div></div><div></div><div></div><div></div></div>	8/10
Palate	<div><div></div><div></div><div></div><div></div><div></div></div>	4/5
Overall	<div><div></div><div></div><div></div><div></div><div></div></div>	17/20

## Yelp



**Yasmin S.**  
 Palo Alto, CA  
 20 friends  
 88 reviews  
 67 photos  
 Elite '18

6/9/2018  
 Cute little ice cream shop with not so ordinary flavors. I sampled the roasted strawberry ice cream and strawberry & kumquat sorbet. Both were good, but I opted for a more interesting mix. The rose pistachio cardamom was nice, though the cardamom overpowered the rose.

[Share review](#)  
[Embed review](#)  
[Compliment](#)

At almost \$5 a scoop, I won't be a regular, but I do appreciate the thoughtful variety of unique flavors offered here.

## Amazon


**R. Suem**

☆☆☆☆☆ **Defective and customer service never responded.**  
 February 27, 2017  
 Color: Matte Black | Configuration: Piano | **Verified Purchase**

The piano is great and the concept is wonderful. The last 3 keys on the far left side are stuck together. I contacted customer service right away and they never got back with me. Really suck. We'll keep it but buyer beware and hope you don't need any customer service.

26 people found this helpful

Figure 4: Example reviews from the data sources considered.