Modeling Hotel Quality Belief in Natural Language Reviews

Evan Shieh
Department of Computer Science
eshieh@stanford.edu

Alex Zamoshchin
Department of Computer Science
alexzam@stanford.edu

1 Abstract

For the average traveler, finding trustworthy hotel reviews online is a tricky task - individual reviews inevitably suffer from bias, and the highest rated review isn't necessarily the most informative. Our research project focuses on modeling how exactly natural language reviews translate to beliefs about hotel quality as indicated on a 5-point ranking scale. In doing so, we draw inspiration from sentiment analysis, which tackles a tangential problem. Additionally, we leverage differences in the hotel review setting (such as skewed rankings, taglines and multiple rankings) to build more accurate classifiers. In all, we analyze the performance of several approaches (Recursive Neural Networks, Recurrent Neural Networks, Long Short-Term Memory, and Multi-task Deep Learning). Among these, our LSTM model performs the best, with a classification accuracy of 66.1% (across 5 possible outputs) and a RMSE of only 0.7313. We surmise that the LSTM achieves this performance above other models by leveraging mean pooling across all tokens in the sentence, as opposed to weighting importance of tokens by time (Recurrent) or structure (Recursive).

2 Introduction

We are interested in the problem of classifying hotel reviews based on reviews from online users. In this setting, ratings on a scale of 1-5 are assigned to each of five categories ("Overall", "Value", "Location", "Service" and "Cleanliness") and paired with a corresponding user review in natural language. At first blush, this problem domain appears to be quite similar to sentiment analysis, since verbal sentiment shares several parallels with expressing reviews (where low reviews are negative and high reviews are positive). Hence, our initial approach centers around using proven models from sentiment analysis (like Recursive Neural Networks). However, analyzing this approach reveals several key differences in our problem space - namely, that reviews are heavily skewed towards strongly positive, and reviews are often composed as unstructured paragraphs with grammatical abbreviations and vernacular English. Additionally, our problem space lends itself uniquely to multi-task learning, as the categories above are often interdependent and can be learned jointly. In all, we construct the data processing pipeline necessary for preprocessing reviews (such as data sanitization/filtering, and converting reviews to binary trees using the Stanford Parser) and then implement and analyze several models, including Recursive Neural Networks, Recurrent Neural Networks, Long Short-Term Memory, and multi-task learning models using both LSTMs and RNNs.

3 Dataset & Baseline

Our dataset is a *TripAdvisor* review dataset provided in JSON format by UIUC, containing plaintext reviews with corresponding 5-point labels for each of several categories (Location, Sleep Quality, Rooms, Service, Value, Cleanliness and Overall). Reviews are each usually several sentences long. We sanitize reviews by filtering out reviews that are missing categories, or ones that are over 800 characters long. In sum, there are 1,250,059 reviews for 12,000 hotels across the world, before 2011.

Ratings for all categories are heavily skewed towards larger ratings. An example of this for the "Overall" category is displayed in *Fig 1* below:

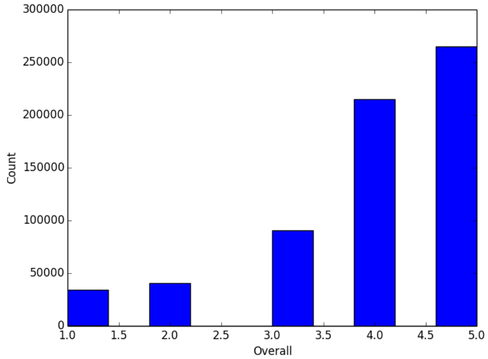

Fig 1. **Histogram of Overall Ratings**

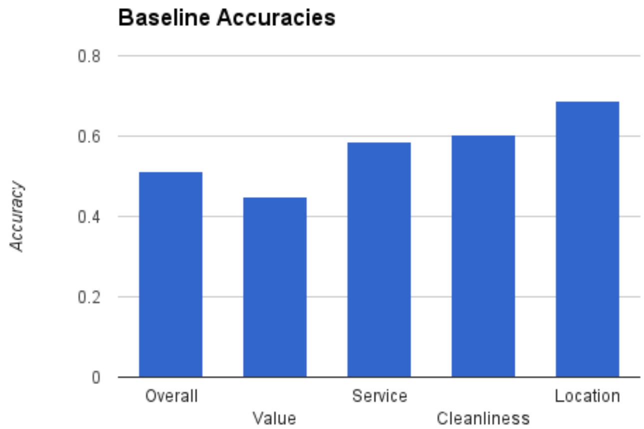

Fig 2. **Baseline Accuracies for All Categories**

Given the large skew of classes, a strong baseline in this investigation is the *argmax* of the priors on classes. Meaning, predicting the most-common class for all sequences achieves the strong baselines for each category seen in *Fig 2* above.

4 Related Work

Our investigation builds on a number of ground-breaking models proposed in the past: Recursive Neural Networks share weights recursively over sequences modeled as binary trees, allowing them to produce predictions using information inherent in the structure of sequences. Recurrent Neural Networks form directed sequences and develop internal states which allow them to model temporal relationships, useful in language modeling and other tasks. Finally, Long Short Term Memory (LSTM) networks build on the success of Recurrent Neural Nets with the addition of memory cells, which will be discussed below. Already having been applied to model sentiment analysis, all three of these models are potentially well-motivated solutions to the task at hand - their advantages and disadvantages in practice, however, will be discussed below.

5 Models

A. Recursive Neural Network

Since the problem of predicting ratings shares many parallels with sentiment analysis, we first explore proven methods in the latter domain. Recursive Neural Networks, when applied to sentiment analysis, operate under the assumption that the sentiment of an overall sentence can be thought of as a recursive composition of the sentiments of its sub-phrases. A similar (and reasonable) argument can be applied to analyzing natural language reviews - logical conjunctions such as “but” will often change the entire meaning of a review as online travelers transition from making qualifying statements to expressing how they feel in reality.

Our model is an extension of the simple one-layer Recursive Neural Network explored in class, but with a key modification. The standard architecture relies on labeling both sentences and sub-phrases, which is not applicable to the review prediction scenario, since users express one ranking (per category) for each review. Hence, during back-propagation, error propagates from only one source (the root node). We confirm (as shown in the results section below) that taking this approach performs better than applying the same label to each sub-phrase.

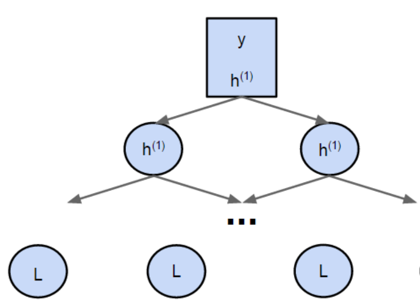

Fig 3. **Depiction of Modified Recursive Neural Net**

Our general model-processing pipeline can be split into three stages:

1. *Filter out invalid reviews.* Our dataset required extensive cleaning, as many data points were either invalid (i.e. containing blank reviews, missing labels, expressed in a different language) or too unwieldy to express as a single binary tree. We drop invalid reviews as well as reviews that are over 800 characters long.
2. *Construct binary trees based on part-of-speech.* We leverage the Stanford Parser to construct standard parse trees based on the English PCFG, and then binarize them for purposes of our analysis.
3. *Perform classification using our Recursive Neural Network architecture.* The modified architecture above is built on top of our assignment in Python.

Directly applying the one-layer architecture as used for sentiment analysis yields a maximum accuracy of 50.9% and a root mean squared error (RMSE) of 1.176 on our dev set, after parameter tuning. By contrast, our modified architecture achieves an improved accuracy of 56.3% with a RMSE of 0.821 on the same dev set.

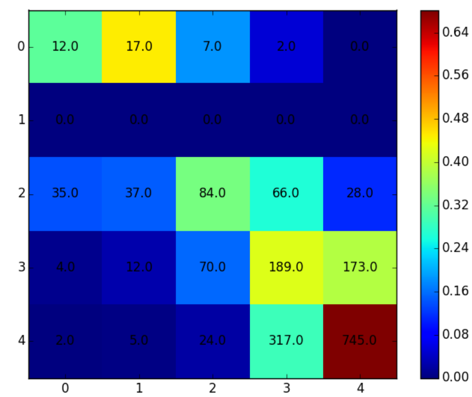

Fig 4. **Confusion Matrix for Recursive Neural Net Predictions**

Additionally, as the confusion matrix in *Fig 4* above shows, our modified architecture predicts a mixture of classifications from 1-star labels to 5-star labels, whereas the original Recursive Neural Network exclusively predicts labels in the 4-star to 5-star range. We surmise that this weakness of the original model is a product of over-amplifying the dataset skew - most labels are already 4 or 5-star reviews, and applying an identical label recursively forces most model parameters towards predicting the mode. Although our model suffers less from this phenomenon, to some extent artifacts exist in our model as well (as evidenced by the fact that our model never predicts 2-star ratings). Observing the training process confirms our hypothesis, since at fewer iterations both Recursive Neural Networks initially predict a mixture of all class labels.

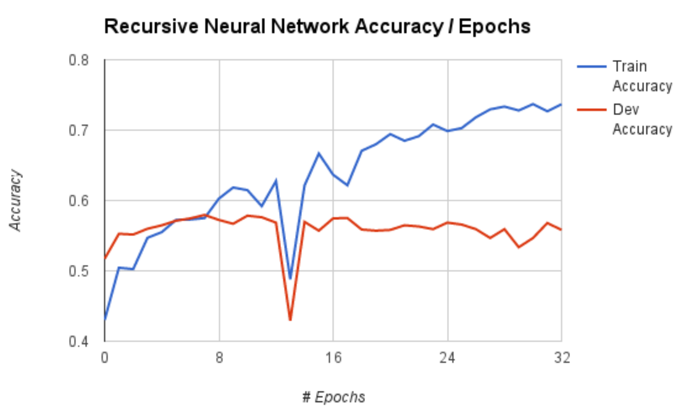

Fig 5. **Recursive Neural Net Classification Accuracy over Time**

Furthermore, plotting classification accuracy for training and dev sets (*Fig 5* above) reveals that the Recursive Neural Network architecture, when applied to this scenario, does not appear to generalize well. Training accuracy increases steadily with more iterations, but dev accuracy tops out at around 56% after just a few training epochs. While this could be a result of the size of the dataset (each set contains roughly 10,000 examples), another plausible explanation is that reviews are not best modeled as trees. The setting of sentiment analysis typically contains well-labeled, structured data in the form of single sentences. By contrast, a qualitative examination of our dataset reviews reveals that reviews generally consist of multiple sentences, often expressed in broken, vernacular English. As a result, the Recursive Neural Net architecture expresses these reviews as long binary trees that might not be appropriate for training purposes, as child nodes may not often be logically related to parent nodes in the casual review setting. Due to this problem (and the amplification of the dataset skew mentioned above), we turn our attention to families of models that make fewer assumptions about the structure of natural language reviews. One of these models is the Recurrent Neural Network.

B. Recurrent Neural Network

Recurrent Neural Networks have grown in popularity for use in language modeling and other tasks. In this paper we examine their use for hotel review modeling. We make a modification (similar to the modification we made for Recursive Neural Nets) to the network structure to account for the lack of fine-grained labeling, outputting only a single rating prediction for each sequence. Similarly, using only the rating of each review as a whole, we back-propagate error across the entire sequence:

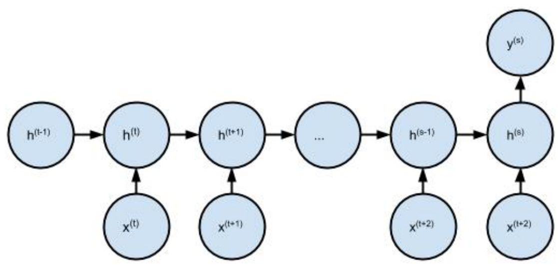

Fig 6. **Depiction of Modified Recurrent Neural Net**

Note that such a model is particularly prone to long-range interactions. This, combined with the fact that we do not split sentences (in order to preserve the entirety of information contained in each review during prediction), means we are left with extremely long sequences with only one label each. The model hence proves to be extremely difficult to converge, achieving only a 49.20% training set accuracy and 49.74% dev set accuracy (with 0.001 learning rate and no weight decay). Moreover, no amount of tuning is able to achieve convergence; we attribute this problem to vanishing or exploding gradients and perform analysis to prove this point:

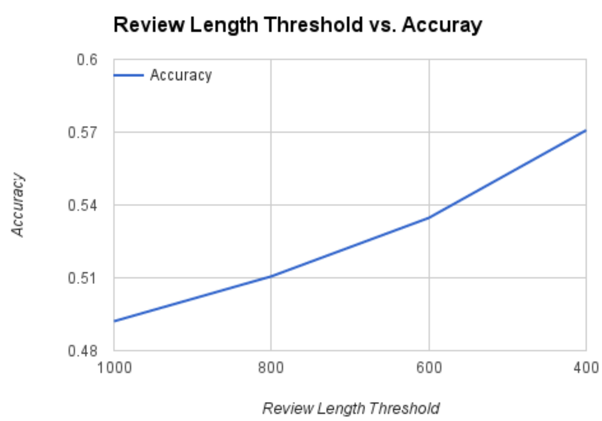

Fig 7. **Results of Recurrent Net on Simplified Train Sets**

Performing thresholding on the lengths of reviews (in characters), we eliminate all reviews below a certain threshold and train on the simplified datasets. We filter out reviews of lengths greater than 200, 400, 600, and 800 characters and analyze our hypothesis. In *Fig 7* above we see the model is able to achieve improved results of 57.08% accuracy on the training and 55.70% accuracy on the dev set, with reviews thresholded at 400 characters. This improvement is indicative of problems of vanishing or exploding gradients and long-range interactions inherent in our model.

More concretely, during back-propagation the gradient is being multiplied a large number of times (for each word in the sequence) by the associated weight matrix, causing the magnitude of the weights to have strong effects on learning. If the weights are small, this leads to vanishing gradients where the gradient signal becomes so small such that learning stops altogether. If the weights are large, this leads to exploding gradients where the gradient signal becomes so large such to cause learning to diverge.

As an initial solution to the problem of exploding gradients discussed above we examine clipping gradients at a magnitude of 5. The hypothesis is that our huge sequences and large-range interactions will result in gradients that are too-large, making fine-grained learning impossible. Using the same learning rate and weight decay parameters as above, we achieve a training accuracy of 53.79% and a dev accuracy of 51.86%, a distinct improvement over the original attempt on the unfiltered dataset.

C. Long Short Term Memory

The analysis above and initial success of gradient clipping motivates the application of Long Short Term Memory (LSTM) networks. LSTM networks are Recurrent Neural Networks with the addition of memory cells - self-contained components that allow the network to learn when to remember and when to forget its hidden states, amongst other possible outcomes. More concretely, a memory cell has four components: an input gate, a self-recurrent connection, a forget gate, and an output gate. These four gates could potentially allow the LSTM to better deal with the long-range interactions present in our problem. Finally, we use an LSTM architecture with mean pooling, in which all hidden cells are directly aggregated into the Softmax layer, allowing the network to again better model long-range interactions.

Training an LSTM network in Theano with 128 word embedding dimension, 0 weight decay, and adadelta optimizer, we achieve the following results:

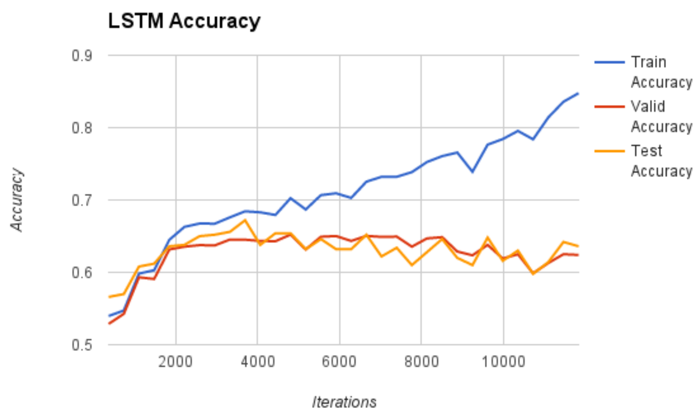

Fig 8. **LSTM Accuracy Results on Train, Dev, and Test Datasets**

Hence, the greatest achieved dev accuracy was 65.21% with a corresponding test accuracy of 65.40% - significantly improved results over those of the RNN implementation above. Furthermore, we achieve a RMSE of 0.7313 on the dev set. The corresponding confusion matrices for these results are displayed below:

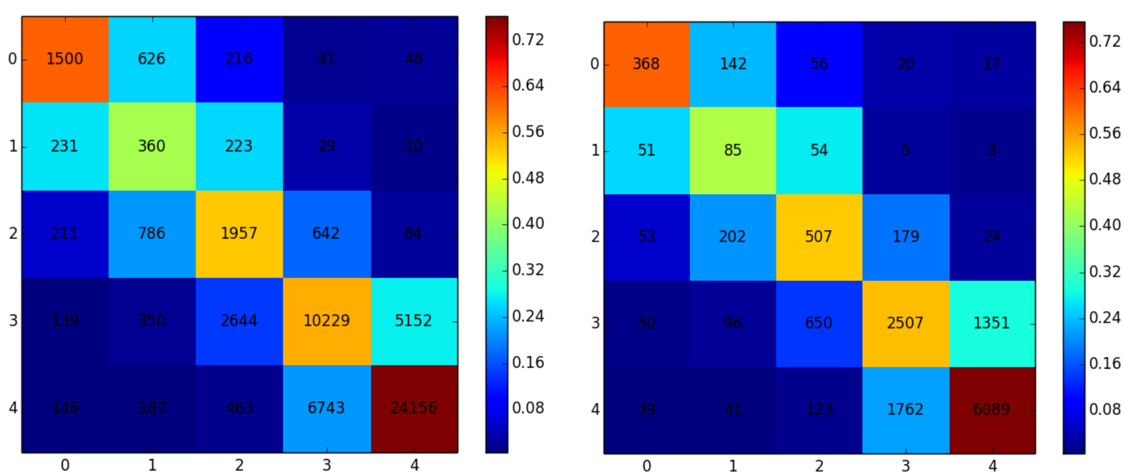

Fig 9. **Train Confusion Matrix (left) and Dev Confusion Matrix (right)**

The model performs relatively well on all labels, especially on extreme rating scores of 1 and 5. It faces some difficulty for reviews falling in the 2-4 range (especially those scoring 2-stars) on both the training and dev sets (perhaps indicating that our model reserves guessing 2-stars only for when it is extremely confident that that is the case). This is similar to the weaknesses of the earlier models we examine, yet indicates that the LSTM is still capable of learning and predicting accurately despite the dataset skew.

Clearly, the memory cells of the LSTM network are at least partially able to improve the modeling long-range interactions inherent in the data. Yet with the improved predictive-power of the network comes a new set of problems: the LSTM model substantially overfits the training set, as evidenced by the large disparity in train and dev set accuracies. In an attempt to address this problem we add L2 regularization and tune the weight decay parameter:

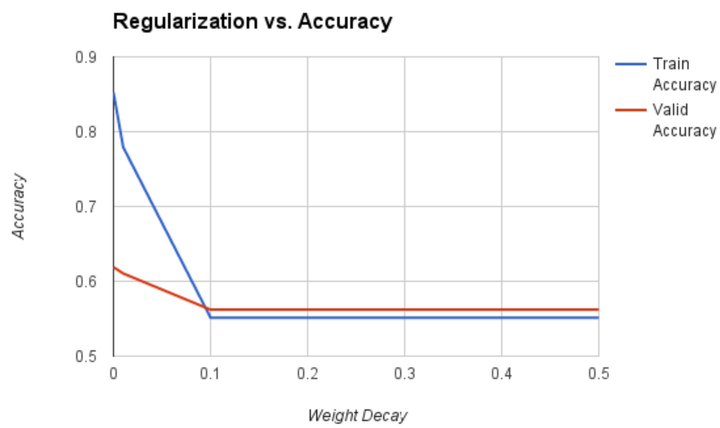

Fig 10. **The Effect of Regularization on Train and Dev Accuracy**

Surprisingly, introducing regularization does reign in the training accuracy but has no effect on improving the dev accuracy. In fact, it seems that even a tiny amount of regularization prevents the model from establishing enough confidence to guess any other label than 5-stars. This may be indicative of the difficulty of classification in a skewed class scenario, and the challenge in improving the generalizability of the LSTM model. We attempt to introduce dropout:

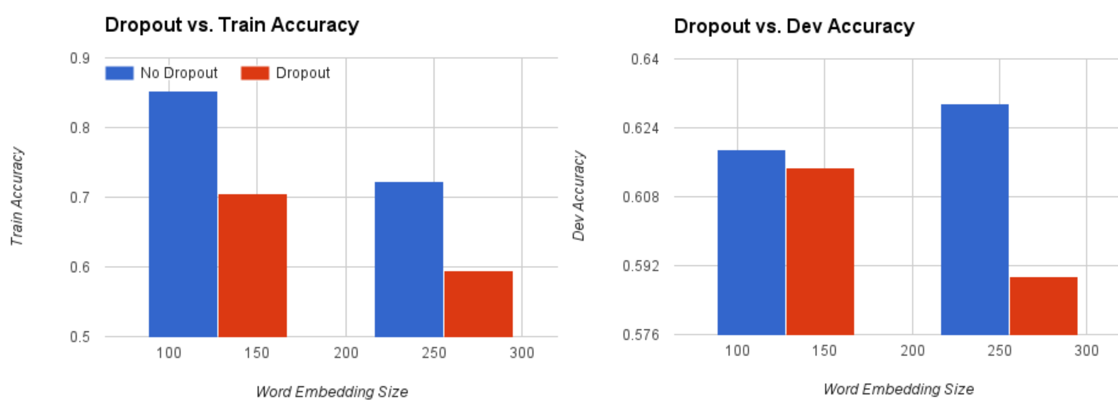

Fig 11. **The Effect of Dropout on Train Accuracy (left) and Dev Accuracy (right)**

We tune dropout at word embedding sizes of both 128 and 256, since the introduction of dropout often-times requires larger models. Dropout proposes to address overfitting by randomly dropping units (with probability 0.5) during training. However, the theoretical improvements in performance are not realized: the model performed better without dropout in all cases, including word embedding sizes of both 128 and 256, and on both train and dev sets.

In a final attempt to address overfitting we attempt to introduce a significantly larger dataset size. Such a dataset was not computationally feasible for most of the investigation but is a useful endeavor in addressing overfitting:

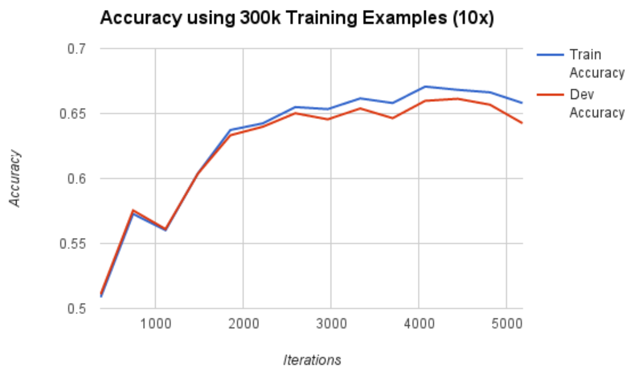

Fig 11. **The Effect of Increased Dataset Size on Train and Dev Accuracies**

The larger dataset size significantly reduces overfitting, almost completely diminishing the disparity between train and dev accuracy. However only slight improvements in the dev and test accuracies are realized: at 66.13% dev accuracy the model achieves a corresponding test set accuracy of 61.80%. This represents the best dev accuracy result achieved in this investigation.

As an additional exploration, we examine how well the model performs using review titles alone, as opposed to both review content and title:

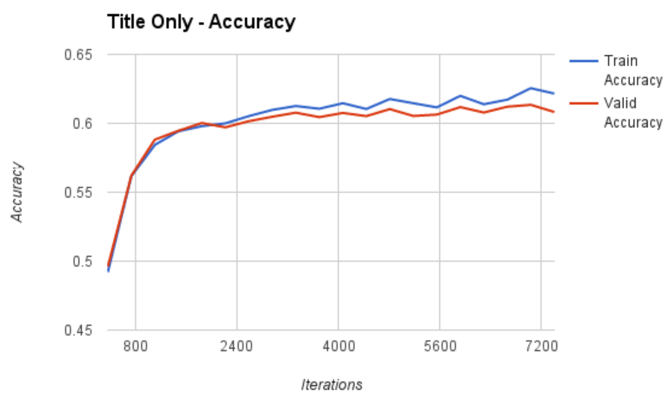

Fig 11. **The Performance of the Model Using Only Title Content**

In *Fig 11* we see the optimal dev accuracy of 61.34% occurs after roughly 7000 iterations, with a corresponding test set accuracy of 60.60%. Although the model was not able to perform as well using only the title alone, it is evident that a substantial part of the improvement over the baseline can be attributed to information gained from the title alone, an interesting result. Moreover, using the title alone also significantly reduces overfitting, perhaps because the decreased quantity of words used results in a lower potential for overfitting.

D. Deep Multitask Learning

Another interesting question we explore is the hypothesis that correlated rating categories can be learned in conjunction with one another. For instance, it is quite possible that reviews that give strongly-positive scores for "Service" are more likely to also rate highly in terms of "Value", and that words used to express these sentiments are overlapping in nature. To approach this question, we turn to the theory of multi-task learning. In broad strokes, multi-task learning acknowledges relatedness among labels by using a shared representation of parameters to learn and predict multiple labels simultaneously (instead of training different classifiers separately). This is especially the case in natural language processing, since we expect many of the earlier layers to be recognizers for broad patterns present throughout the sequences. Hidden layers and word vectors will be shared and updated by a sum of errors from multiple labels as opposed to just one.

More concretely, suppose there are C different categories of labels that we would like to learn simultaneously (C=5 in our example: Overall, Value, Location, Service, Cleanliness). Then, using the Recursive Neural Network framework as an example, we implement multi-task deep learning as follows.

Forward propagation is extended as follows:

$$\hat{y} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_C \end{pmatrix}$$

where each of C probability vectors is appended together to create one output vector. Each probability vector is calculated as follows:

$$\hat{y}_i = softmax\left(W_i^{(s)} h + b_i^{(s)}\right)$$

As seen from the equation above, the weight matrix W and bias vector b vary for each of C label categories, but the hidden layer (and the parameters used to calculate the hidden layer) are shared as parameters across all label categories. In our implementation, we append these separate per-category weight matrices and bias vectors together in order to calculate the output probabilities (of dimension $5 \cdot C$) directly.

Back-propagation is simply modified to include the sum of all errors across the new C-hot truth vector. Our analysis treats errors from all categories with equal importance, but it would not be difficult to modify this error sum to be a weighted sum of categories (treating "Overall" with greater importance, for instance).

$$\delta = \Sigma_{i=1}^{C} (\hat{y}_i - y_i)$$

We apply multitask learning to extend our Recursive Neural Network model across the five categories mentioned above, with varying accuracies as depicted below. Performance on the "Overall" category is comparable to our single-task model, predicting a maximum accuracy of 53% on the dev set.

Fig 12. **Mutli-task RNN Accuracy for Training (left) and Dev (right)**

As *Fig 12* above indicates, the multi-task classification approach appears to benefit the model learning for the "Overall" category (in blue) the most strongly, as classification accuracy increases by the most across epochs for both the training and dev set (the relative differences between lines are due to varying dataset skew). This would indicate that the "Overall" category is most highly interrelated to the other rankings categories. Examining the RMSE (which is arguably more informative a metric and less biased by the skew of rating labels) provides further support for this hypothesis:

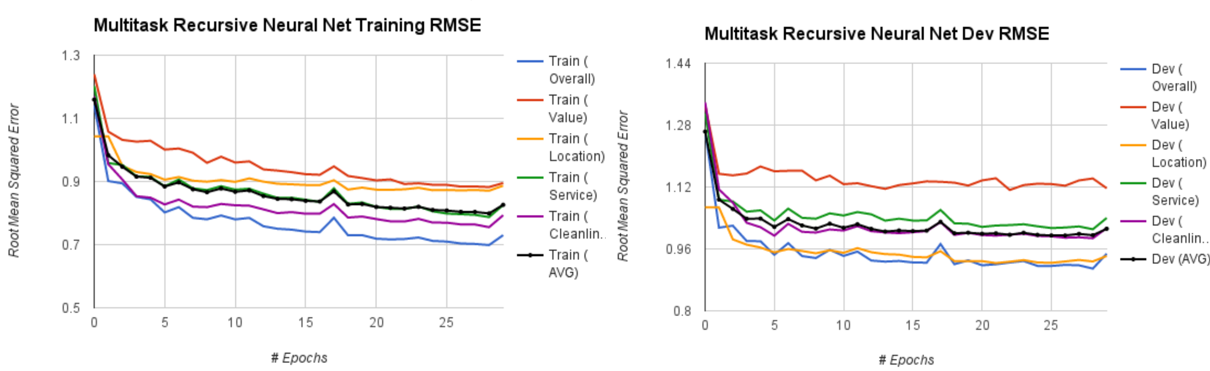

Fig 13. **Mutli-task RNN RMSE for Training (left) and Dev (right)**

The above graphs show that the RMSE for the "Overall" category (in blue) experiences the greatest decrease compared across all categories, and reliably remains the lowest as the number of training epochs increases.

Interestingly, the RMSE for the "Location" category (in yellow) also experiences a comparatively large decrease on the dev set. This could indicate that "Location" is perhaps the ranking most highly correlated to other rankings. If true, this would be an interesting trend to note, as other categories in theory should not depend on the location of the hotel.

Similarly, we also hope to extend the results achieved by the LSTM network to the multi-task scenario. We build on the LSTM network to address classification of all C tasks in the manner discussed above, the results for which are seen below:

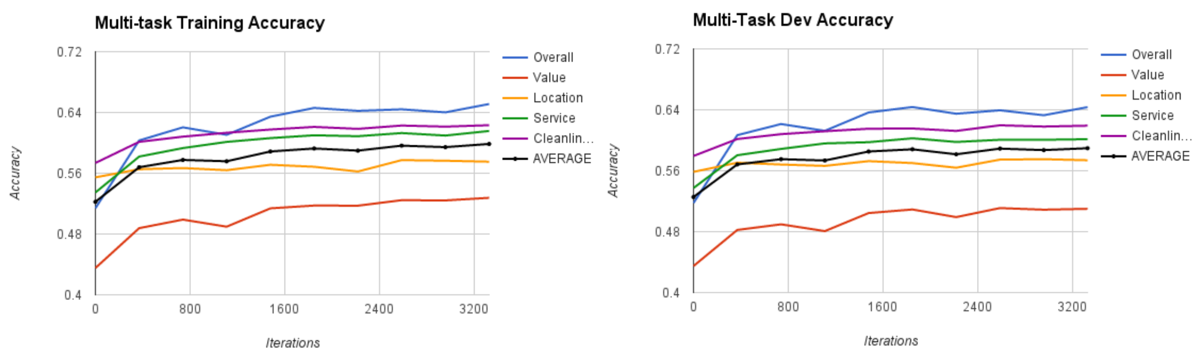

Fig 14. **Mutli-task LSTM Accuracy for Training (left) and Dev (right)**

Note that the best average validation accuracy is achieved after about 2600 iterations:

	Train	Dev	Test
Average Accuracy	59.63%	68.91%	57.20%

Table 1. **Mutli-task LSTM Average Accuracy Results**

On a more granular basis, we see the following test accuracies for each category:

	Overall	Value	Location	Service	Cleanliness
LSTM Accuracy	62.40%	47.40%	54.00%	60.00%	62.20%
Baseline Accuracy	51.40%	44.84%	68.62%	58.47%	60.33%

Table 2. **Mutli-task LSTM Test Accuracy Results for All Categories In Comparison to Baseline**

Comparison of the multi-task model's test set accuracy on the "Overall" rating of 62.40% with LSTM's prior result of 65.40% indicates that no substantial improvement was achieved via modeling as a multi-task LSTM task. Moreover, the model performed better on less-heavily skewed categories (Overall and Value) than on more-heavily skewed categories (Location), indicating that perhaps the model experienced increased performance on harder tasks at the expense of easier ones. Therefore, such a method may still be beneficial if the objective function of the investigation is to optimize the average accuracy of *all* categories.

6 Conclusions and Future Work

Our work outlines a general framework for analyzing natural language, multi-label online review data. Of the various models we examine - Recurrent Neural Nets, Recursive Neural Nets, and Long Short-Term Memory - we find that LSTMs are best suited for the classification task, achieving a maximum classification accuracy of 66.1% (across a label space of 5 discrete outputs) and an RMSE of 0.7313 (with the maximum error of 4). Parts of the other two models still perform well, but are less suited for our classification task for a variety of reasons. Recursive Neural Nets make the strong assumption that language semantics are structured recursively, which we found not entirely true for casual, abbreviated online reviews that span multiple sentences. Meanwhile, Recurrent Neural Nets suffer from long-range interactions and exploding gradients for long sequences of reviews. We also analyzed both LSTM and the Recursive Neural Net models with multi-task learning and find that strong correlations across labels do exist. With respect to multi-task learning, we find that while overall accuracy is not improved, accuracy on harder tasks (i.e. less skewed categories) is improved at the cost of accuracy on easier tasks. This actually posits multi-task learning as an effective strategy for learning against skewed datasets. Lastly, we learn on simply review titles alone using our best model (the LSTM) and find that these titles also hold surprising power in predicting rating.

Further work for our project will involve applying the (relatively reliable) models we have implemented to answer pertinent questions about online reviews. For instance, we would like to consider how reviews vary across geography, time and price range. Does a model trained on reviews from the United States have predictive power in classifying reviews from the UK? How does the price range of the hotel influence how predictable reviews for the "Value" category are? Given our work in choosing and tuning models to accurately predict hotel reviews, we are interested in turning our attention towards queries about natural language that can arguably only be tackled using statistical machine learning.

7 References

Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Bergstra, James, Goodfellow, Ian, Bergeron, Arnaud, Bouchard, Nicolas, and Bengio, Yoshua. Theano: new features and speed improvements. NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2012.

Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for Scientific Computing Conference (SciPy), June 2010.

Graves, Alex. Supervised sequence labelling with recurrent neural networks. Vol. 385. Springer, 2012.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. Neural computation, 12(10), 2451-2471.

Socher, Richard; Lin, Cliff; Ng, Andrew Y.; Manning, Christopher D. "Parsing Natural Scenes and Natural Language with Recursive Neural Networks". The 28th International Conference on Machine Learning (ICML 2011).