Learn To Rate Fine Food

Jiacheng Mo, Lu Bian and Yixin Tang
Stanford University, US

***Abstract*— We investigate a food review dataset from Amazon with more than 500000 instances, and utilize information from the data set such as the text review, score, helpfulness, etc. Instead of the traditional word representation using frequency, we use skip-gram to train our own word vectors using the pre-trained GloVe Twitter word vector as the initialized value. We also use recursive parsing tree to train the vector of the whole sentence with the input of word vectors. After that, we use neural network methods to classify our review text to different scores. We mainly research and compare the performance of Gated Recurrent Unit Network (GRU) and Convolutional Neural Network (CNN) on our data. After tuning the hyper parameters, we get our best classifier as a bi-directional GRU. We also build a Long Short Term Memory Model (LSTM) for text generation, which is able to generate text for each score level. Then we build a recommendation system based on Latent Factor Model, with Stochastic Gradient Descent, and recommend 10 items to selected users. Finally, we use softmax regression to visualize the most important words for a certain score, and design a spam review binary classification based on the helpfulness scores of the reviews.**

I. INTRODUCTION AND RELATED WORK

In recent years, food reviews have become increasingly popular on social media. People are posting their reviews or comments on Facebook, Yelp, Twitter, etc. When selecting local restaurants or food, people also tend to make their decisions based on these reviews. Hence, it is important for both restaurants and individuals to quickly get the information and score of a food item or restaurant from thousands of reviews. It is also beneficial for some platform to provide different customers with their personal recommendations.

McAuley and Leskovec [1] have done related work on recommending beer, wine, food, and movies, where they built a latent factor recommendation system that explicitly accounts for each user's level of experience. Here, however, we built a less sophisticated latent factor recommendation system, with the intent to achieve reasonable accuracy.

Marx and Yellin-Flaherty [2] have done related work on sentiment analysis of unstructured online reviews using GRU and LSTM model, where they incorporated a relatively small data set. Here, we extend the discussion to a larger data set, and we have used LSTM model for review generation that has never been done before.

II. DATASETS AND FEATURES

Our analysis focus on an Amazon food review database consisting of $568,454$ instances. The data set contains the texts of reviews, scores and helpfulness. In more detail,

- Product ID: the product that this review is for. The total number of items that are rated is $62,279$. And the average number of reviews for an item is 9.2.

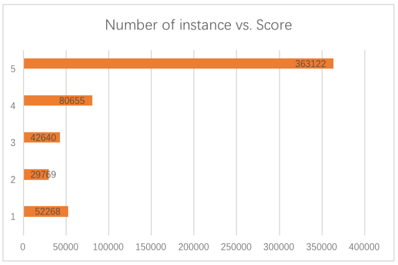


Fig. 1: Distribution of scores of reviews.

- User ID: the user who wrote this review. The total number of users is $178,554$. So the average number of reviews that each user gave is 3.2.
- Text: the main content of the review.
- Helpfulness: number of people who find this review helpful.
- Score: the score this user gave to the food item.

The histogram in Fig. 1 shows that the number of positive reviews is greater than the negative ones. Also, the number of reviews with score 5 is much greater than other reviews. This shows that customers are more likely to give a review when they feel either very satisfied or disappointed.

III. METHODS

A. Word to Vectors

1) Skip-Gram: Instead of using the traditional count-based representation of words, we train the word vectors from our own data. First, we download the *GloVe Global Vectors for Word Representation* which was trained on the crawl data of Twitter(2B tweets, 1.2M vocab) [3]. We can visualize the vector representation by finding the nearest neighborhood of a certain word through L2 measure. For example, The 7 nearest words of ”frog” are ”rana”, ”litoria”, etc.

However, we do not directly use this representation for our final representation of words. Instead, we just use this as an initializer of our self-trained Word to Vector model. We train our word vectors by skip-gram model, which uses the method of predicting the surrounding words in a window of certain length. The objective function is to maximize the log probability of any context word given the current center word:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} log(P(w(t+j)|w(t)))$$

where θ represents all variables we optimize and $p(w_{t+j}|w_t)$ is the probability:

$$p(o|c) = \frac{\exp(u_0^T v_c)}{\sum_{w=1}^{W} \exp(u_w^T v_c)}$$

where o is the outside word vector and c is the inside word vector. Every word in this model has two vectors.

2) Phrase Vectors: Now we have got the vector representation of each single word. But our aim is to classify the entire review (in sentences). Hence we need to derive a method to vectorize the sentences. A natural approach is to average or concatenate all word vectors in a given sentence. It turns out that this works well and can be very convenient to implement like the Fig. 2.

We are not satisfied so far since neither concatenation nor averaging treats each word equally and neglects the relationship between words and the structure of the sentence. Hence we use a recursive parsing tree to get our sentence vector. Concretely, for every representation of two candidate children, we calculate the semantic representation if the two nodes are merged, as well as a score showing how plausible the new node would be. The score of a tree is computed by the sum of the parsing decision scores at each node. We train our tree by maximizing the max-margin parsing. [4]

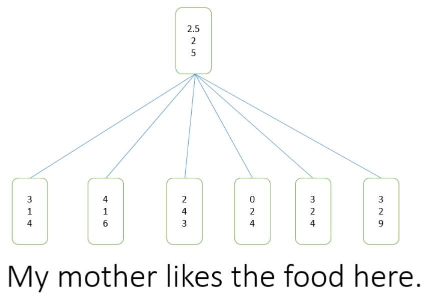


Fig. 2: Sentence Representation by Averaging

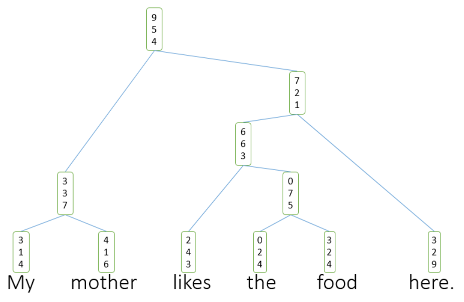


Fig. 3: Sentence Representation by Parsing Tree

$$J = \sum_i s(x_i, y_i) - \max_{y \in A(x_i)} (s(x_i, y) + \Delta(y, y_i))$$

Fig. 3 shows the illustration of this method on the same sentence.

B. Neural Network Methods

1) Gated Recurrent Units: Our first task is to group reviews into 5 classes. The main method we use involve models that are neural network based and are all implemented in tensorflow. We use logistic regression as our baseline model and try to add some layers and more complicated structures to reach a higher accuracy. The state-of-the-art model we implemented for classification is the Bidirectional Gated Recurrent Units (GRU).

We know that the standard Recurrent Neural Network computes hidden layer at the next time step directly by:

$$h_t = f(W^{hh} h_{t-1} + W^{hx} x_t)$$

In GRU, we first compute an update gate according to the current input word vector and the hidden state:

$$z_t = \sigma(W^z x_t + U^z h_{t-1})$$

then we compute the reset gate with different weights:

$$r_t = \sigma(W^r x_t + U^r h_{t-1})$$

and obtain the new memory content and the final memory:

$$\tilde{h}_t = tanh(W x_t + r_t \circ U h_{t-1})$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

Fig. 4 shows the framework of our bidirectional GRU. The difference between the standard GRU and our bidirectional GRU is that for each hidden layer, the information flows not only from the left but also from the right. the concrete formulas of our 3-layers bidirectional GRU are listed below.

$$\begin{aligned}
&Right \quad direction \quad \vec{h}_t^{(i)}: \\
&\vec{z}_t^{(i)} = \sigma(\vec{W}_{(i)}^{(z)} x_t^i + \vec{U}_{(i)}^{(r)} h_{t-1}^{(i)}) \\
&\vec{r}_t^{(i)} = \sigma(\vec{W}_{(i)}^{(r)} x_t^i + \vec{U}_{(i)}^{(r)} h_{t-1}^{(i)}) \\
&\tilde{\vec{h}}_t^{(i)} = tanh(\vec{W}_{(i)} x_t + r_t \circ \vec{U}_{(i)} h_{t-1}) \\
&\vec{h}_t^{(i)} = z_t^{(i)} \circ h_{t-1}^{(i)} + (1 - z_t^{(i)}) \circ h_t^{\tilde{(i)}} \\
\\
&Left \quad direction \quad \overleftarrow{h}_t^{(i)}: \\
&\overleftarrow{z}_t^{(i)} = \sigma(\overleftarrow{W}_{(i)}^{(z)} x_t^i + \overleftarrow{U}_{(i)}^{(r)} h_{t-1}^{(i)}) \\
&\overleftarrow{r}_t^{(i)} = \sigma(\overleftarrow{W}_{(i)}^{(r)} x_t^i + \overleftarrow{U}_{(i)}^{(r)} h_{t-1}^{(i)}) \\
&\tilde{h}_t^{(i)} = tanh(\overleftarrow{W}_{(i)} x_t + r_t \circ \overleftarrow{U}_{(i)} h_{t-1}) \\
&\overleftarrow{h}_t^{(i)} = z_t^{(i)} \circ h_{t-1}^{(i)} + (1 - z_t^{(i)}) \circ h_t^{\tilde{(i)}} \\
\\
&Output \\
&y_t = softmax(U[\vec{h}_t^{(top)}; \overleftarrow{h}_t^{(top)}] + c)
\end{aligned} \tag{1}$$

By feeding a review into each x, we allow each batch review to influence each other's prediction, an idea that may seem counterintuitive since each review is independent. However, different reviews may have a certain connection with the same food or restaurant. Only through this structure can we explore the deeply connected information of reviews. After we fetch y_i from GRU, we use cross entropy loss for the network and train the weights.

Beyond the 3-layers GRU network, we also obtain good results from a 2-layers GRU, 2-layers Convolutional Neural Network, and 3-layers Convolutional Neural Network. We summarize the accuracy later to compare these models with different hyper-parameters.

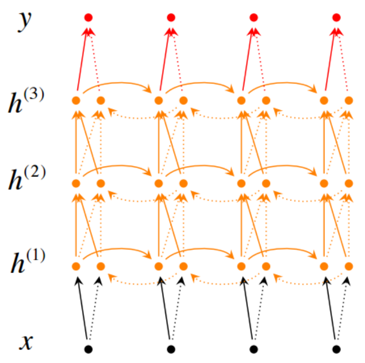


Fig. 4: 3 Layers GRU

2) Long Short-Term Memory (LSTM): After the classification, we want our model to automatically generate some reviews. Concretely, we first specify a score representing the rank of the reviews (like 5 means the best and 0 means the worst), and then feed this score to the neural network. We want our neural network to generate the review corresponding to the specified score.

The model we use is LSTM, which is also a modified version of recurrent neural network but with some reset and forget state. This method selects the most probable word at each time, depending on which class we are in. Fig. 5 gives us a brief structure of LSTM.

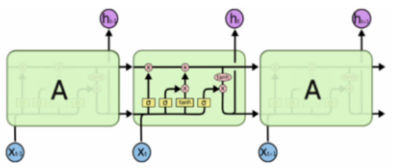


Fig. 5: LSTM for Generation

Our algorithm for generation can be simplified as:

```
$Given \quad a \quad random \quad h_0$
    $For \quad i \quad = 1, 2, 3, 4....$
        $h_{i+1} = Function \quad above(h_i, x_i)$
        $Choose \quad the \quad largest \quad probability \quad class$
        $of \quad h_{i+1} \quad to \quad be \quad x_{i+1}$
                                                        (2)
```

C. Recommendation System

We use Latent Factor Model to recommend food items to a user. As shown in Fig. 8, the sparse user-item utility matrix R can be decomposed to a user-factor matrix P and a item-factor matrix Q. In our model, we choose the number of factors to be 300. After decomposition, we use Stochastic Gradient Descent (SGD) to decrease the loss, and find the final matrices P and Q, whose product is closest to R. Denote the R_{iu} of the matrix R the rating given by user u to item i. The total error is:

$$E = \sum_{(i,u) is rated} (R_{iu} - q_i p_u^T)^2 + \lambda(\sum_u ||p_u||_2^2 + \sum_i ||p_i||_2^2)$$

Denote ϵ_{iu} the derivative of the error E with respect to R_{iu}, then

$$\epsilon_{iu} = R_{iu} - q_i p_u^T$$

And the update equations for q_i and p_u in SGD are:

$$q_i \leftarrow q_i + \eta(\epsilon_{ui} p_u - \lambda q_i)$$

$$p_u \leftarrow p_u + \eta(\epsilon_{ui} q_i - \lambda p_u)$$

After 40 iterations of SGD, we obtain the final P and Q. Then, we predict the score the user will rate an unrated item, and recommend 10 items of the highest scores to the user.

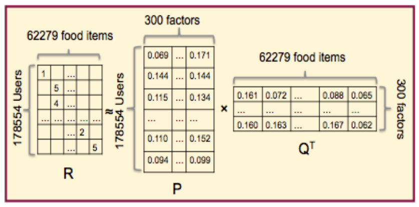


Fig. 6: Latent Factor Model

D. Features Visualization and Spam Reviews

In this part, we use the traditional count based word representation and compare the results with the previous approach. We utilize count vectorizer followed by term frequency–inverse document frequency (tf-idf) [5] transformation to transform the text into word tokens. Count vectorizer converts a collection of text documents to a matrix of token counts and produces a sparse representation of the counts. The following tf-idf transformation converts the count matrix to a normalized tf-idf representation. In information retrieval, using tf-idf, instead of the raw frequencies of occurrence, can scale down the impact of words that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus. For example, words such as ”the”, ”is”, etc. can appear in any text in general, and thus are offset by the frequency of the word in the corpus in tf-idf representation.

For finding the words that contribute the most to each of the five scores of the food reviews, first we use feature selection to select 1000 best informative words out of the 10000 features (from tf-idf). The next step is to build a softmax regression model on these features and to fit between words, x and scores, y. In softmax regression, we maximize the log-likelihood:

$$l(\theta) = \sum_{i=1}^{m} \log \prod_{l=1}^{k} \left(\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \right)^{1\{y^{(i)}=l\}}$$

Finally, we choose the largest 50 coefficients in θ_i of the probability of each class y_i, and get the corresponding words as the most important words for the score.

Since the relationship between a word and the helpfulness of the corresponding review is not as clear as the relationship between a word and the score of its corresponding food item, we define an ”helpful” review to have a helpfulness score greater or equal to 4, where as an ”useless” review has 0 as helpfulness score. Thus, this becomes a binary classification problem, with truncated data set that does not contain reviews with 1, 2, or 3 helpfulness score. We fit the data to softmax regression model, quadratic discriminant analysis model [6], as well as the decision tree model [7], where we varied the regularization parameter and plotted training and cross-validation accuracy versus regularization parameter.

IV. RESULTS

A. Neural Network Methods

1) GRU: The results of GRU and CNN model and the loss sequence of 3-layers GRU are summarized in the table below:

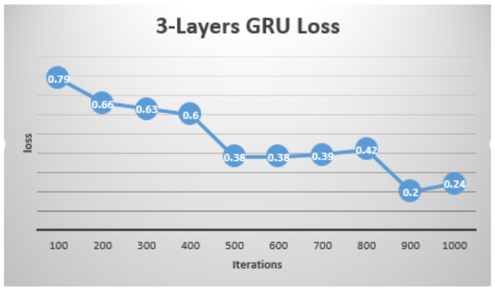
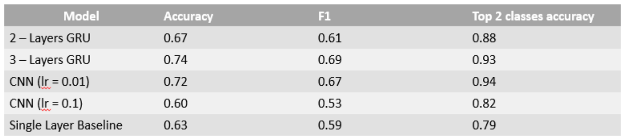


Fig. 7: 3-Layers GRU Loss Function

We list part of our experiments as above. We start with the vanilla neural network of a single layer and its test accuracy is 0.63 which is not bad when we have 5 different



Model	Accuracy	F1	Top 2 classes accuracy
2 – Layers GRU	0.67	0.61	0.88
3 – Layers GRU	0.74	0.69	0.93
CNN (lr = 0.01)	0.72	0.67	0.94
CNN (lr = 0.1)	0.60	0.53	0.82
Single Layer Baseline	0.63	0.59	0.79

Fig. 8: Accuracy Table

classes. Then we use convolutional neural network and tune the learning rate. From the table we find that the learning rate 0.01 is more suitable to reach a better minimum. Finally, we use GRU with two and three layers respectively to train our data. We find 3-Layers GRU can reach a higher accuracy and its top 2 classes (scores 4 and 5) accuracy is 0.93. This is meaningful, because usually we only need a broad idea of whether this food is good or not, instead of a explicit score.

2) LSTM: Here is an example text generated from our LSTM of score 5 food review. We show the process of picking the highest probability from the last hidden layer and fetching into the next input layer.

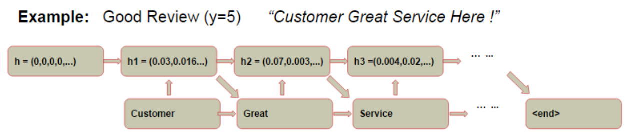
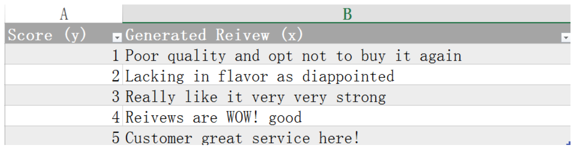


Fig. 9: Level 5 Generation

With the the number of iteration increases, the generated sentence makes more sense to us, including the punctuation. For each class, we select some reasonable generated sentence to present here.



Score (y)	Generated Reivew (x)
1	Poor quality and opt not to buy it again
2	Lacking in flavor as diappointed
3	Really like it very very strong
4	Reivews are WOW! good
5	Customer great service here!

Fig. 10: Generated Review

B. Recommendation System

In the SGD process, we have changed the learning rate η to get the lowest convergence error after 40 iterations(see figure 11). We finally decide to use 0.02 to get the lowest error. (Another way is to decrease the learning rate as iteration grows.)

We can get the recommendation for an user by finding the largest scores in the row corresponding to this user in the utility matrix R. There is a sample recommendation for this user shown in figure 12. However, the recommendation result is not as good as we thought. There exists a set of products being recommended to many different users. The reason behind this problem may be the fact that our model does not include an item bias, so the scores of some items may be much greater than others for all the users. These

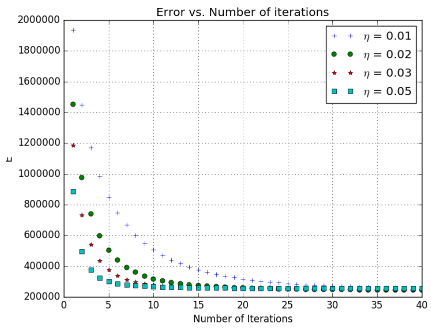
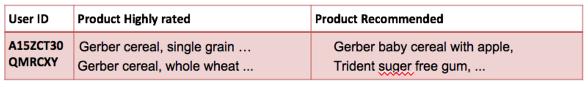


Fig. 11: Error decreases as iteration grows, different learning rate

items, such as trident gums, are much more likely to be recommended than others.



User ID	Product Highly rated	Product Recommended
A15ZCT30 QMRCXY	Gerber cereal, single grain ... Gerber cereal, whole wheat ...	Gerber baby cereal with apple, Trident suger free gum, ...

Fig. 12: Recommendation result example

C. Features Visualization and Spam Reviews

After fitting the data to the softmax regression, we compute 50 words that have the largest coefficient and thus are the most significant words for each score. We organize these words into tag clouds(Fig. 13), where the size of each word is based on the relative value of the coefficient.

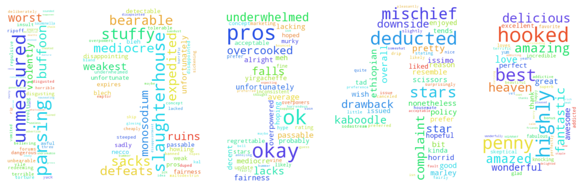


Fig. 13: Tag cloud for scores from 1 to 5

First we fit the data to softmax regression model, and find that the training error as well as cross-validation error are 0.44 and 0.42 respectively. We notice that the softmax regression model failed, possibly due to the non-linearity of the frequency of a word versus the score of the food item.

We obtain decent training and cross validation accuracy when fitting our data to both QDA and decision tree model. We vary the regularization parameter and plotted training and cross-validation accuracy versus regularization parameter(Fig. 14).

V. CONCLUSIONS AND FUTURE DIRECTIONS

We use multiple neural network models to classify our text. Our results show that bi-directional GRU has the best performance with high efficiency. Another advantage of our GRU is that it is immune to extremely unbalanced data. During our experiments, we find that when the distribution of data is not uniform. For example, when most food items

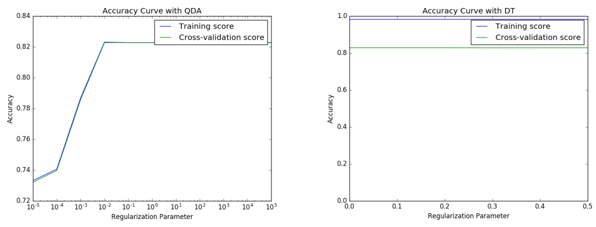


Fig. 14: Cross validation accuracy curve for QDA and for DT

are in level 2, most classifiers will just assign any input to the majority class to reach a high accuracy. However, GRU can still learn it well and not stick to the majority. A natural generalization of our model is to apply it on different review tasks, say, Yelp, Twitter, Walmart, Ikea, etc. This requires our model to automatically choose the hyper parameter and structure, in order to accept the input under different context. This will be our new direction to design a more robust classifier which can be used upon any kinds of customer review.

Our recommendation system is based on the basic latent factor model. We have reached a relatively low error by changing the learning rate in SGD process, but the recommendation result is not very satisfying. This model can be further improved by adding some terms, such as, user/item bias, implicit feedback, temporal dynamics, user-associated attributes, and confidence level [8]. Also, if time is allowed, we can develop a better recommendation system. A common way is to use hybrid method and build a system that has many levels filtering [8]. For instance, we can add a global effect filtering before latent factor model, and after that, add a collaborative filtering(user-user or item-item).

Softmax regression succeeded in selecting words for a specific score, but it failed in modeling the helpfulness of a word. For the latter task, both QDA and decision tree model fit the data well, mainly due to the non-linearity of the frequency of a word versus the score of the food item.

REFERENCES

[1] McAuley, and Leskovec *From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews.* Proceedings of the 22nd international conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2013.

[2] Marx, and Yellin-Flaherty *Aspect Specific Sentiment Analysis of Unstructured Online Reviews*

[3] GloVe Twitter Crawl : http://nlp.stanford.edu/projects/glove/

[4] Taskar et al. 2004 Learning Structured Prediction Models: A Large Margin Approach

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

[6] Friedman et al. *The elements of statistical learning* Springer, Berlin: Springer series in statistics, 2001.

[7] Breiman et al. *Classification and regression trees.* CRC press, 1984.

[8] Jure Leskovec et al. *Mining of Massive Datasets* Cambridge University Press, 2012.