From Movie Reviews to Restaurants Recommendation

Xing Margaret FU, Xiaocheng LI *
(SUID: chengli1, xingfu)

June 8, 2015

Abstract

In this project, we first examine word vector representation of movie reviews and conduct sentiment analysis on this dataset. We compare word vectors learned from different language models and their performance on predicting review sentiment. With these exploratory results, we transfer the learning task to prediction of recommendation on another restaurant review dataset. The basic recommendation algorithm is built on matrix factorization of user-restaurant ratings. Adding features from text reviews substantially improves the prediction. Our model outperforms the state-of-art model (McAuley and Leskovec, 2013) in mean squared error.

1 Introduction

The prevalence of social media has encouraged the spreading of online opinion through blogs, social networks and e-commerce websites. The proliferation of reviews, ratings and recommendations has motivated the interest of sentiment analysis. The recent breakthrough in word vector representation of text data drives the progress in many natural language processing aspects. Vector-based models encode words in a continuous space that maintains word distance by their semantic similarities. After setting up the word vector space by semantic distance, it is much accurate for sentiment classification on large text datasets. Furthermore, incorporating contextual information of user-product reviews also improves recommendation accuracy. In order to well integrate this information, we first explore different methods of representation and their performance on a simpler task – sentiment analysis. We then use the exploratory results onto another Yelp restaurant review dataset and predict recommendation for users.

2 Movie Reviews Sentiment Analysis

We use an IMDB movie review dataset from Kaggle challenge (https://www.kaggle.com/c/word2vec-nlp-tutorial/data) for sentiment analysis. It includes 25,000 labeled movie reviews and 50,000 reviews without any rating labels. The sentiment of reviews is binary, meaning the IMDB rating < 5 results in a sentiment score of 0, and rating ≥ 7 have a sentiment score of 1. We divide the labeled dataset into training (16,500) and development sets (8,500), apply Bag of Words, Word2Vec and GloVe to train feature vectors, and use random forest, K-means and CNN for sentiment classification.

2.1 Models for Sentiment Analysis

2.1.1 Representation: Bag of Words

The bag-of-words model represents each text as a vector of the number of occurrences each word appears. It learns the vocabulary from all the documents. This representation disregards grammar and word order, but it does keep word multiplicity.

*Authors in alphabetic order.

In the IMDB data, we have a very large number of reviews, which will give us a large vocabulary. We first remove punctuation, numbers and stop words such as "a", "and", "is" and "the". The stop words are from Python Natural Language Toolkit. Also, to limit the size of the feature vectors, we should choose some maximum vocabulary size. Below, we use the 5000 most frequent words after excluding stop words.

2.1.2 Representation: Word2Vec And GloVe

We learned word vector space representation in details both in class and in assignments. Here we use Python library Genism's implementation of word2vec model with skip-gram and hierarchical softmax, to train our word vectors.

In order to better capture the global corpus statistics, such as word-word co-occurrence matrix, we apply GloVe to train global word vectors. Similar to Word2Vec, GloVe represents words by input and output vectors, and models the probability of two words being contexts. Different than Word2Vec, GloVe uses the co-occurrence matrix as responses and a weighted least square model to train the parameters. The cost function of GloVe model is

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2.$$

Here X_{ij} is the (i, j) entry of the co-occurrence matrix denoting the number of co-occurrence of word$_i$ and word$_j$; w, $\tilde{w}$, b and $\tilde{b}$ are input and output word vector and intercept terms as in Word2Vec. The weight function $f(X_{ij})$ is defined as,

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise.} \end{cases}$$

We use the implementation package from http://nlp.stanford.edu/projects/glove/ to train our word vectors on the movie review dataset.

2.1.3 K-means

The default feature construction from words to reviews is to create average vectors for each review paragraph. With these numeric training features, we apply random forest to classify review sentiment. As we learned from Assignment 1, this does not yield good performance on sentiment classification.

Since word vectors reflect semantic and syntactic information of words, we consider applying K-means clustering to all the word vectors to group together semantically related words. Then borrowing the idea from Bag of Words, we now convert reviews into feature vectors using bag of clusters. We use random forest classifier from scikit-learn to train and predict sentiment of reviews.

2.1.4 CNN

One disadvantage of K-means is that it loses word order in a document. To overcome this, we employ the convolutional neural network into the sentiment analysis. Based on the well-trained word vectors, we construct one convolutional layer which combines the features of consecutive 2, 3, 4 and 5 words. After that, we do max-pooling with regard to the convolutional layer and make the predictions. From the experiments on word vectors, we know that our movie document is too small for further word vectors tuning, so we keep all the word vectors static during the training.

2.2 Experiments

2.2.1 Word Vector Evaluation

We evaluate the two sets of word vectors learning using Word2Vec and GloVe on the movie review training dataset. We use Google's standard evaluation questions (http://word2vec.googlecode.com/svn/trunk/questions-words.txt) and results are shown below. Since our corpus is quite small and only targeted on movie reviews,

the overall performance of word vectors is poor because the corpus does not include enough information needed for certain semantic tasks. PCA on word vectors also shows limited separation of word clusters.

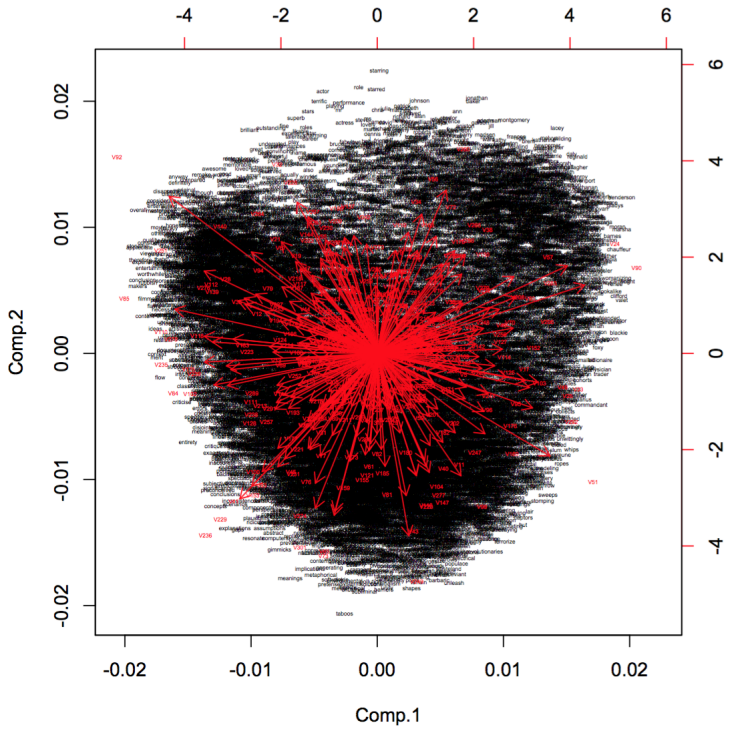

Figure 1: Trained Word Vectors PCA Plot

	Word2Vec	GloVe	GoogleNews WordVecs
capital-common-countries	24.444%	23.162%	24.737%
capital-world	24.000%	10.000%	14.992%
currency	0.000%	0.000%	12.151%
city-in-state	1.190%	1.215%	13.952%
family	58.684%	57.381%	84.585%
gram1-adjective-to-adverb	4.630%	2.688%	28.528%
gram2-opposite	1.667%	2.769%	42.734%
gram3-comparative	27.742%	32.583%	90.841%
gram4-superlative	12.648%	14.021%	87.344%
gram5-present-participle	21.846%	25.985%	78.125%
gram6-nationality-adjective	26.496%	15.297%	21.923%
gram7-past-tense	21.780%	17.857%	65.962%
gram8-plural	16.270%	11.593%	89.865%
gram9-plural-verbs	38.587%	24.217%	67.931%

Table 1: Question and Answer Accuracy for Our Word Vectors

2.2.2 Sentiment Classification

For sentiment analysis, we compare the results using Bag of Words, Word2Vec with average features, Word2Vec with K-means and Word2Vec with CNN. All of these representations except for Word2Vec with CNN use random forest with 100 trees for classification. The baseline Bag of Word method reaches 84% accuracy, which is higher than any word vector representations. Google News word vectors have lowest accuracy. CNN keeps word order in a document and better harness the meaningful word vectors, which results in the best performance.

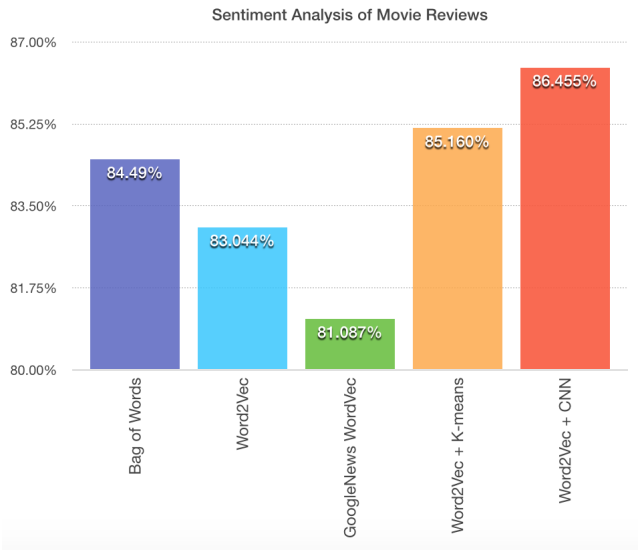

Figure 2: Performance of Different Models on Movie Review Sentiment Analysis

3 Restaurants Recommendation

In previous section, we investigate on the problem of movie review sentiment analysis and achieve good performance with CNN classifier based on word vectors. After that, we hope to explore the restaurant review data which has different structures. First, besides the review text, each thread of restaurant rating is linked with a specific user and a specific restaurant. Second, the task is different in that we no longer want to do prediction on a review text but we hope to predict the rating based on a user ID and a restaurant ID. So in this recommendation system settings, the review texts no longer serve as predictor but help us to do feature extraction for each user and each restaurant.

3.1 Data Description

For our project, we will work on Yelp Business Rating dataset (https://www.kaggle.com/c/yelp-recsys-2013/). The data contains 229907 reviews from 45981 reviewers for 11537 restaurants. To begin with, we divided the dataset into train, dev and test according to the ratio $3:1:1$. All the training and hyper parameter tuning are done on the train and dev set, while all the performance reported below are evaluated on test set. It is noticeable that we only extract feature (for users and restaurants) from review text on training set. When it comes to rating prediction on test set, we only need user ID and restaurant ID and disregard the review text. This is a more realistic setting in that when Yelp attempt to predict the user's rating toward certain restaurant, it only knows the user ID and restaurant ID but doesn't know the user's review toward the restaurant beforehand.

Following the convention of [6] and [8], we evaluate the performance of the models via mean square error. For all predicted ratings p_i and actual ratings a_i, we calculate the mean square error as follow.

$$\text{MSE} = \frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}$$

Here n is the total number of samples in test set.

3.2 The Model

3.2.1 Traditional Approach of Matrix Factorization (As Baseline)

Traditional approaches to resolve recommendation problem always regard it as a matrix completion problem. Assume the rating matrix as $M \in \mathcal{R}^{m \times n}$, where m is the number of users and n is the number of restaurants. We know a small proportion of the ratings, namely some entries $M_{(i,j)}$ when $(i,j) \in \Omega$ and hope to predict those unknowns. To treat the problem as a matrix completion problem, we factorize the matrix M into two low rank matrices $U \in \mathcal{R}^{m \times p}$ and $R \in \mathcal{R}^{p \times n}$ and sole the following optimization problem:

$$\min_{U \in \mathcal{R}^{m \times p}, R \in \mathcal{R}^{p \times n}} \sum_{(i,j) \in \Omega} (M_{i,j} - (UR)_{i,j})^2 + \lambda \left(\|U\|_F^2 + \|R\|_F^2 \right).$$

Essentially, what we obtain after minimizing the objective functions are feature vectors u_i (each row of U) for each users and r_j (each column of R) for each restaurant. For those unknown entries, we could simply predict them with the inner product of u_i and r_j.

In our project, we implement the matrix factorization model (via stochastic gradient descent due to the large dataset) and treat the result as a performance baseline. To compare the performance, we also employ the result in paper [6], where they builded a Bayesian latent topic models for the review text. Our idea is quite similar to theirs in that both models aim to extract user/restaurant feature from review text, but our model is simpler and enjoys much better performance.

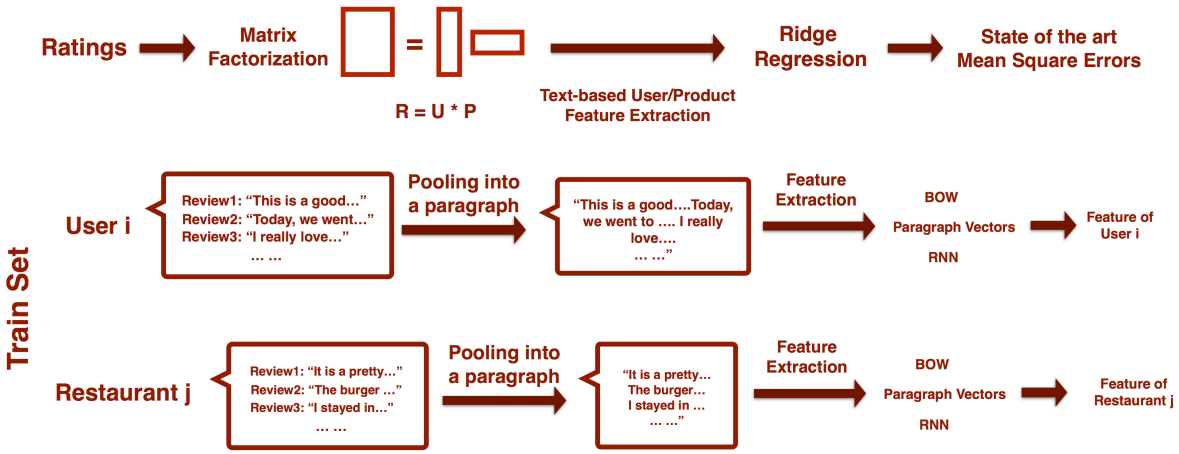

Figure 3: Our Model

3.2.2 Feature Extraction from Review Text

The idea for feature extraction is quite simple. For a specific user, we first pool together all the reviews he/she has written and form a big review "paragraph" for him/her. Then we try to summarize a feature vector for this user based on his/her "paragraph". To emphasize, we only look at the reviews on train set and disregard those on dev/test set. To obtain feature vector for a big "paragraph", we come up with several ideas as below:

- Bag of words (BOW): We use the occurrence of words to represent the paragraph.
- Paragraph Vectors (PV): In the paper [7], they proposed a model to learn a feature vector for a paragraph. The model is quite similar to the word2vec model.
- Recurrent neural network (RNN): RNN is a good tool to summarize paragraph vectors from word vectors.

We could replicate the same process toward each restaurant. And after all these, we would get a feature vector $\tilde{u}_i$ for each user and $\tilde{r}_j$ for each restaurant.

3.2.3 Combine Matrix Factorization and Review Text Feature

To combine the features u_i and r_j purely learnt from ratings and the features $\tilde{u}_i$ and $\tilde{r}_j$ purely learnt from review text, we proposed the linear models below to predict the rating matrix entry $M_{i,j}$.

$$M_{i,j} \approx \mu + u_i^T r_j + w_u \tilde{u}_i + w_r \tilde{r}_j$$

The above prediction function could be interpreted as a linear regression based on the result of matrix factorization (MF). We implement the MF model as previous work and obtain a baseline prediction $\mu + u_i^T r_j$. Then we use $w_u \tilde{u}_i + w_r \tilde{r}_j$ to fit the residue $M_{i,j} - \mu - u_i^T r_j$. To avoid overfitting, we employ ridge regression to train w_u and w_r. To clarify, $\tilde{u}_i$ and $\tilde{r}_j$ are obtained on the train set via unsupervised learning and they remain fixed from then on. And μ ,u_i, r_j, w_u and w_r are either trained or tuned on dev set.

3.3 Result

3.3.1 Mean Square Error

The performance of our model is reported in Table 2. So far, among the three ideas to extract features from review text, we have implemented two of them: bag of words and paragraph vectors (RNN on the way). The mean square errors are significantly reduced with the newly introduced text-based features. While the BOW feature enjoys the lowest MSE, the paragraph vector (PV) achieves a slightly weaker performance with far smaller feature dimensions.

Models	Mean Square Error
Mean Guess	1.473
MF	1.288
MF + Topic Models	1.224 (See [6])
MF+BOW (6000 dim)	**1.164** (Ours)
MF+ PV (50 dim)	1.178 (Ours)

Table 2: Mean Square Errors of Recommended Ratings

3.3.2 Result Analysis

By looking into details on the regression weights, we get an understanding of why the text-based feature could boost the performance of the matrix factorization model. In Table 3, we summarize all the BOW feature with absolute weights larger than 0.005.

Positive Words	Weights	Negative Words	Weights
amazing	0.0077	manager	-0.0093
work	0.0071	downtown	-0.0079
scottsdale	0.0070	bread	-0.0075
friendly	0.0066	meal	-0.0065
yelp	0.0063	point	-0.0064
recommend	0.0061	waitress	-0.0061
dessert	0.0060	burger	-0.0051
worth	0.0060	walmart	-0.0051
selection	0.0052		
wife	0.0051		

Table 3: Regression Weights for BOW features

As we can see from the table, these words with large weights meaningfully represents certain characteristics of a user/restaurant. For example, a restaurant located in "Scottsdale" tends to have a higher rating while a restaurant serve "burgers" at "downtown" is likely to have a lower rating. If the manager and waitress are mentioned in the previous review of a restaurant, it might have some problem in costumer service. More interesting, if a user once used the word "wife" in his review, he must get married and would have an intention for high rating just because he always goes to restaurant with his wife/family. These features are much easier to extract from review history but quite challenging to obtain from the rating matrix as latent variables.This justifies why our approach could increase the performance of matrix factorization model.

4 Conclusion and Future Work

In this project, we explore how to extract meaningful information from text data on social media. On the movie review data, we build a CNN based on word vectors to achieve a good sentiment analysis performance. On the restaurant rating data, we introduce text-based feature to rating prediction and improve the recommendation quality. Our project is far from finished but rather opens up many new possibilities in text-based information extraction, especially in terms of the restaurant recommendation task.

- Other feature extraction approaches: In restaurant/user feature extraction, we only implemented BOW and PV as feature extraction techniques. RNN or other models might be more effective here.
- Effectiveness over different data domains: The text-based feature extraction techniques in the project works well for Movie and Restaurant data. However, do they enjoy the same advantage with different dataset such as Amazon product review data? With larger numbers of review, user and products, the performance of the model awaits more experiments.
- More complicated models: In our rating prediction model, we didn't take into consideration the interaction between user feature $\tilde{u}_i$ and restaurant feature $\tilde{r}_j$ but only treated them as independent linear factors. More complicated models could be explored here, such as the inner product of the two features like MF model or including interaction terms into the regression.

References

[1] Kaggle: Bag of Words Meets Bags of Popcorn, https://www.kaggle.com/c/word2vec-nlp-tutorial

[2] Pennington, Jeffrey, et al. (2014) GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar.

[3] Kim, Yoon, (2014) Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar.

[4] Mikolov, Tomas, et al. (2010) Recurrent neural network based language model. *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, Makuhari, Chiba, Japan, September 26-30, 2010.

[5] Irsoy, Ozan, and Claire, Cardie. (2014) Opinion mining with deep recurrent neural networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

[6] McAuley, Julian, and Jure, Leskovec. (2013) Hidden factors and hidden topics: understanding rating dimensions with review text. *Proceedings of the 7th ACM conference on Recommender systems, ACM.*

[7] Mikolov, Tomas, et al. (2013) Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems.*

[8] Ting, Jason, et al. (2013). "Yelp Recommendation System."