Recommendation Systems for Movies with Sentiment Analysis using Neural Networks

Colby Wise
Columbia University
cjw2165@columbia.edu

Michael Alvarino
Columbia University
maa2282@columbia.edu

Richard Dewey
Columbia University
rld2126@columbia.edu

Abstract

In this research paper we apply the methodology outlined in the arXiv working paper: "Joint Deep Modeling of Users and Items Using Reviews for Recommendation" for rating prediction of movies. The approach used in this paper models users and items jointly using review text in two cooperative neural networks. We have extended their model by implementing a different architecture and objective function. We plan to further extend it by testing different hyperparameters and architectures.

1. Introduction

Our initial milestone recreated the DeepCoNN model outline in the JDM paper using texts reviews for movies from the Amazon Instant Video dataset. The results of this recreated model will serve as a baseline for model comparison and evaluation. For the milestone we made a number of small improvements to the model

Source Code:: Joint Deep Modeling of Users and Items Using Reviews

1.1. Related Work

The literature on recommendation systems is deep and until recently has traditionally focused on well-known matrix factorization algorithms, such as collaborative filtering, as popularized by Netflix and Spotify [Lee, et. al., 2012]. The benefits of collaborative filtering is that it is relatively easy to implement and computationally efficient given most similarity measure i.e. cosine similarity is matrix multiplication. Conversely, the draw backs in real-world applications have significant implications: namely CF suffers from the "cold-start" problem in that it requires user ratings to make predictions. For instance, for new users without prior history it's hard to predict ratings/recommendations thus models have poor generalization [Balakrishnan, Dixit, 2016]

Recent research has focused on using Deep Learning, specifically convolutional and recurrent neural networks to help solve the generalization problem. Specifically, we will be focusing on expanding the Deep Cooperative Neural Networks (Deep-CoNN) model thus our primary reference will be the original [Zheng, Noroozi, et. al. 2017] paper. The authors do not provide their code online, but other have recreated it for video games reviews here Joint and Deep. Additional related papers that we referenced for constructing the Deep portion of our network include DeepFM [Guo, Tang, et. al. 2017] which similar to our model, combines a matrix factorization with a CNN architecture; and TransNets [Catherine and Cohen. 2017], which extends DeepCoNN to examples where the users review is not available.

It should be noted that our original proposal was to build and improved model of Google's Wide and Deep model. We spent a considerable amount of time going down this path, but ultimately came to the conclusion that we could not source the high quality categorical data needed for training the wide part of the model. We decided to switch to the DeepConn model due to it's improved performance and the availability of high quality data for training, although the model is more complex and will likely be computationally more expensive to train.

1.2. Problem Formulation

In order to make better movie recommendations how can we more accurately predict a users rating for an unseen movie based on what the user has previously seen? Furthermore, can we improve generalization of our model when information on a users past movie ratings is limited?

These two questions are the crux of our problem. Prior to deep learning, standard approaches for recommendation systems ($RecSys$) used collaborative filtering which relies on decomposing users, items (i.e. movies), and ratings into latent feature matrices. Then the weights of these matrices are used to predict a rating a user would give for an item. One common method includes using the cosine similarity measure between all pairs of movies that users have rated:

Where,

m_i and m_j refer to movie1/movie2 and denote vectors of ratings from users have rated both movies:

$$\textbf{sim}(m_i,m_j) = cos(\theta) = \frac{\vec{m_i} \bullet \vec{m_j}}{||\vec{m_i}||_2 \times ||\vec{m_j}||_2}$$

This yields a movie-to-movie similarity matrix of dimensions $M \times M$ with ones along the diagonal. Thus, the predicted rating for movie $m_2$for user1 would be calculated using similarity measures between (m_2, m_1) and (m_2, m_3) weighted by the respective ratings for m_1 and m_3.

From the above formulation we can clearly see the major disadvantage of this approach: sparsity. When ratings are limited the movie-to-movie matrix is mainly zeros thus limiting predictive ability. In recent years a researcher have attempted to work around this limitation by using text data that users write after watching movies. This text data is used as input for training neural networks and offers additional insights to simple numerical ratings or categorical features.

The DeepCoNN model formulation that we'll focus on in this research uses two jointly modeled neural networks. The first network Net_i uses all of the text reviews for a given user. The second network Net_p uses all of the text reviews for a particular product (movie).

1.3. Data and Methodology

By modifying a code base found online Join and Deep we were able to replicate one of the base-case scenarios for the DeepCoNN model. The original code base was incomplete and not well suited for our data, thus our reconfiguring of the DeepCoNN model presented a number of technical challenges. First, we used GloVe (50) to transform our reviews into a vector representation. GloVe returns a matrix representation for each word, so to get the data into the same form as that which is presented in the paper, we take the mean of each column in the GloVe word matrix. Thus for each review we have N words in the review and 50 columns for an N x 50 matrix.

The second technical issue to work through is combining the two networks into a single loss function. This aspect is key to this model and others designed to combine the results of two (or more) neural networks and train them jointly with a common loss function. A number of approaches exist, but as a first step, we tested the baseline implementation of the model, by taking the dot product of the output from the fully connected layer of Net_i and output from the fully connected of Net_p. This method is straightforward to implement, but suffers from the fact that doesn't capture higher level interactions between features. For our final project we plan to implement more robust methods for jointly modeling two neural networks, including using a factorization machine. The the CNN architecture described in the DeepCoNN paper was built using Keras and many of the Keras native methods.

1.4. Architecture

The first layer of the network items (movies are represented as matrices of word embeddings as discussed above. The next layers are standard to CNN's including the convolution, max pooling and fully connected layers. We use ReLU activations for each convolutional layer as is standard in the literature. The output of each network is joined by taking the dot product of X_u and Y_i the output vectors of the two neural networks. The approach explained in the paper introduces a shared layer, which allows us to map the two vectors into the same feature space. This is accomplished by concatenating the two vectors X_u and Y_i and then estimating a factorization machine on the resulting vector Z. As described in the paper, we optimize the model using RMSprop, which is an adaptive version of gradient descent that controls the step size with respect to the absolute value of the gradient. The current implementation does not use regularization, however we plan to implement regularization into our final project.

1.5. Preliminary Results

Our preliminary work toward the milestone consisted of reconstructing aspects of the code as highlighted above to model text review data. We confirmed the results reported in the Joint Deep Modeling paper for the DeepCoNN-DP implementation and extended the original DeepCoNN by using alternative hyperparameters of the original architecture that are detailed in Exhibit 1.

1.6. Further Research

For our final project we intended to test various methods for building a shared layer between in the two networks, including a factorization machine. We also intended to test other hyperparameter setting and potentially different architectures including an LSTM structure. Finally, as a robustness check, we intend to test our model on other datasets, including the amazon TV and video games datasets. Both of these datasets have similarly structured text reviews and should be amenable to the DeppCoNN predictive model.

Model	Losses	Training Time
DeepCoNN-DP[1]		
Stand Alone FM		

Exhibit 1. Model Summary

2. Diagrams

Model Diagram

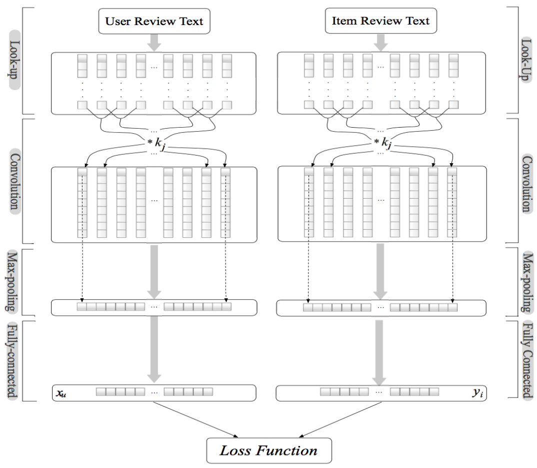

Figure 1: The architecture of the proposed model

2.1. References

[1] Zheng, Lei, Noroozi, Vahid and Yu, Philip. Joint Deep Modeling of Users and Items Using Reviews for Recommendations. arXiv working paper. June 2017

[2] Rendle, Steffan. Factorization Machines. ICDM 2010

[3] Catherine, Rose and Cohen, William. TransNets: Learning to Transform for Recommendation. arXiv working paper. June 2017

[4] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In KDD, pages 19?28, 2009.

[5] P. Baldi and P. J. Sadowski. Understanding dropout. In NIPS, pages 2814?2822, 2013.

[6] Y. Bengio, L. Yao, G. Alain, and P. Vincent. Generalized denoising auto-encoders as generative models. In NIPS, pages 899?907, 2013.

[7] C. M. Bishop. Pattern Recognition and Machine Learning. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.