B.Tech Project Report - Semester 7

Recommendation Systems

under the guidance of

(Prof. Sudeshna Sarkar)

Submitted By:
Abhik Naskar
17CS30001

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
November, 2020

Contents

x-x------------------x-x

Introduction

A recommender system, or a recommendation system (sometimes replacing 'system' with a synonym such as platform or engine), is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. They are primarily used in commercial applications such as e-commerce shopping, YouTube video suggestions etc.

Motivation:

The motivation behind this project is to perform sentiment analysis on user reviews and thereby analysing the nature of review whether positive or negative sentiment and ultimately predict rating for an unobserved item by a user.

Work Done:

Worked on various models using deep learning methodologies and text preprocessing techniques to analyse the reviews and predict the sentiment. Traditionally, recommender systems are based on methods such as clustering, nearest neighbor and matrix factorization. However, in recent years, deep learning has yielded tremendous success across multiple domains, from image recognition to natural language processing. Recommender systems have also benefited from deep learning's success. In fact, today's state-of-the-art recommender systems such as those at Youtube and Amazon are powered by complex deep learning systems, and less so on traditional methods.

Plan of Work

To improve the quality of recommendation and to perform better sentiment analysis on user/item reviews we can undertake a neural unified review recommendation system with cross attention which we shall discuss in due course of time. A simplified model for the above discussed methodologies has been implemented and the complete model is to be implemented in further course of action.

Sentiment Analysis - Amazon Fine Food Reviews

Sentiment analysis (also known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

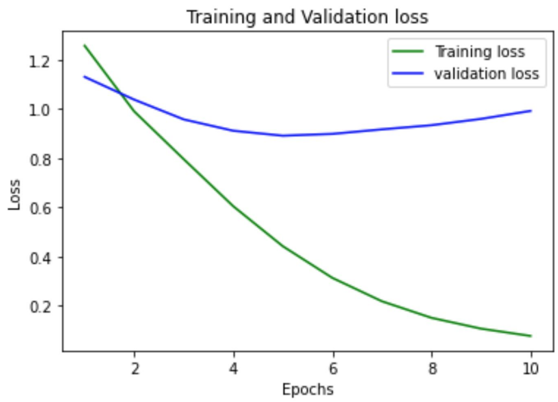
<u>**Steps for Sentiment Analysis:**</u>

A. Text Pre-processing:
   a. Replaced regex with ' ' apart from english alphabet.
   b. Convert all words to lowercase english alphabets
   c. Removed common english stop words such as and, the, or etc.
   d. Performed Stemming on texts to retain the root/base word

B. Word Embeddings: To convert text to meaningful numbers for proper sentiment analysis and to be fed into a neural network model, we use word embeddings.
   a. Store index of words in a corpus as numeric representation.
   b. TF-IDF vectorizer
   c. Word2vec model
   d. Glove Model

PS : The text pre-processing method is kept uniform across all the models discussed later.

Model 1 : Neural Network Model using single hidden layers followed by an output layer

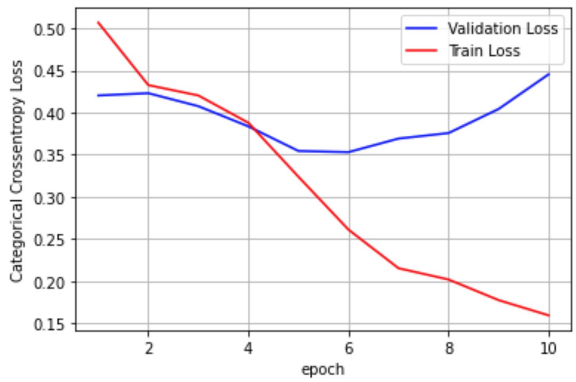



```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_26 (Dense)             (None, 64)                666880
_________________________________________________________________
dense_27 (Dense)             (None, 5)                 325
=================================================================
Total params: 667,205
Trainable params: 667,205
Non-trainable params: 0
_________________________________________________________________
None
```

1. The output layer has 5 output units since the scores vary from 1-5.
2. In this model we used a TF-IDF vectorizer as a word embedding model.
3. Performance accuracy : 67.5%
4. From the above plots, the model seems to overfit and thereby such low accuracy on unseen data sets.
5. From further analysis, we annotate scores > 3 as ‘Positive’ review and scores<=3 as ‘Negative’.

Model 2 : LSTM with single output Dense Layer

An LSTM model consists of a **cell state** and **forget gate** as its architecture. The forget gate helps the model to analyse which words to remember and what to forget thus giving a proper sentiment analysis for individual texts.




```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_11 (Embedding)     (None, 600, 32)           96032
_________________________________________________________________
lstm_13 (LSTM)               (None, 100)               53200
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 101
=================================================================
Total params: 149,333
Trainable params: 149,333
Non-trainable params: 0
_________________________________________________________________
None
```

- As evident from the model summary we can see that we used an Embedding layer which outputs vector representation of words in 32 dimensions as visible from output shape. This vector representation in n-dimensional space provides a clear evidence of sentiment by grouping the vectors with same sentiment together(cosine similarity).
- The model has an accuracy of ~82%.
- Results:

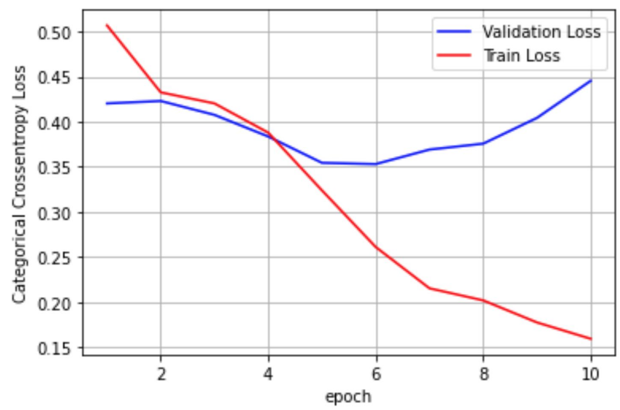
  a. Sentence : “Not as Advertised”

  Output : “Negative review with 48% accuracy”

Thus we can see the model doesn’t perform quite well on unseen data which implies a hint for overfitting.

Model 3 : multiple LSTM layers along with dropout with ratio 0.25 and 0.5 respectively

---




```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_11 (Embedding)     (None, 600, 32)           96032
_________________________________________________________________
lstm_13 (LSTM)               (None, 100)               53200
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 101
=================================================================
Total params: 149,333
Trainable params: 149,333
Non-trainable params: 0
_________________________________________________________________
None
```

1. In this model we improve the previous model by utilising the regularisation parameters such as dropout.
2. Performance Accuracy : 84.42%
3. Example:
   - Input : “not as advertised”
   - Output: “negative review with ~56% accuracy”
4. The model performed quite better than the previous with predicting actual sentiment although with quite less accuracy.

Model 4 : Sentiment Analysis using Deep CNN

Dataset : IMDB movies review dataset

```
Model: "functional_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 50)]         0
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 50, 300)      673500      input_3[0][0]
__________________________________________________________________________________________________
conv1d_10 (Conv1D)              (None, 49, 200)      120200      embedding_2[0][0]
__________________________________________________________________________________________________
conv1d_11 (Conv1D)              (None, 48, 200)      180200      embedding_2[0][0]
__________________________________________________________________________________________________
conv1d_12 (Conv1D)              (None, 47, 200)      240200      embedding_2[0][0]
__________________________________________________________________________________________________
conv1d_13 (Conv1D)              (None, 46, 200)      300200      embedding_2[0][0]
__________________________________________________________________________________________________
conv1d_14 (Conv1D)              (None, 45, 200)      360200      embedding_2[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_10 (Global (None, 200)          0           conv1d_10[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_11 (Global (None, 200)          0           conv1d_11[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_12 (Global (None, 200)          0           conv1d_12[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_13 (Global (None, 200)          0           conv1d_13[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_14 (Global (None, 200)          0           conv1d_14[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 1000)         0           global_max_pooling1d_10[0][0]
                                                                 global_max_pooling1d_11[0][0]
                                                                 global_max_pooling1d_12[0][0]
                                                                 global_max_pooling1d_13[0][0]
                                                                 global_max_pooling1d_14[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 1000)         0           concatenate_2[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 128)          128128      dropout_4[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 128)          0           dense_4[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 2)            258         dropout_5[0][0]
==================================================================================================
Total params: 2,002,886
Trainable params: 1,329,386
Non-trainable params: 673,500
```

1. We Performed Text Processing and Word Embeddings using the pre-trained Google news word2vec model
2. In the neural network architecture we applied five different sized filters in the CNN model each outputs 200 filters.
3. Output : To predict score(positive(1) or negative(0) on the unseen data(Test data here).
4. The model achieved an accuracy of 74.67% on the test dataset.

NEURAL UNIFIED REVIEW RECOMMENDATION WITH CROSS ATTENTION

- Approaches for exploiting the reviews information can be categorised into two major methods:

1. **Document level methods:**
    a. In this method, we concatenate all the reviews of a user/item into a long single document and then analyse the features of the document.
    b. DeepCoNN uses parallel convolutional filters for semantic analysis of the features representing documents.
    c. Since all the reviews are given equal weightage, this model alone is not practically feasible.
2. **Review level methods:**
    a. Since different reviews are of different informativeness, we model each review first and then focus on more important reviews for user/item representation.
    b. Attention mechanism is used to explore the usefulness of reviews at review level after obtaining feature reviews by CNN as discussed above.

- We design **document-level encoder** for document level representation of user/item.
- For review level we use **review encoder** to extract each review-feature of words and then use a **user/item encoder** to learn review level representation of user/item.
- Finally, we fuse two types of feature together to obtain the final comprehensive representations of users/items.

- **Goal :** The goal is to predict the rating that a user would score towards an unobserved item.

Cross Attention Module

- For the above three encoders discussed above, we apply cross attention module to retain the important words and reviews.
- We represent the user/item as a low valued dimensional vector as embeddings.
- We utilize Multi layer perceptron to generate cross attention vectors for each user/item.

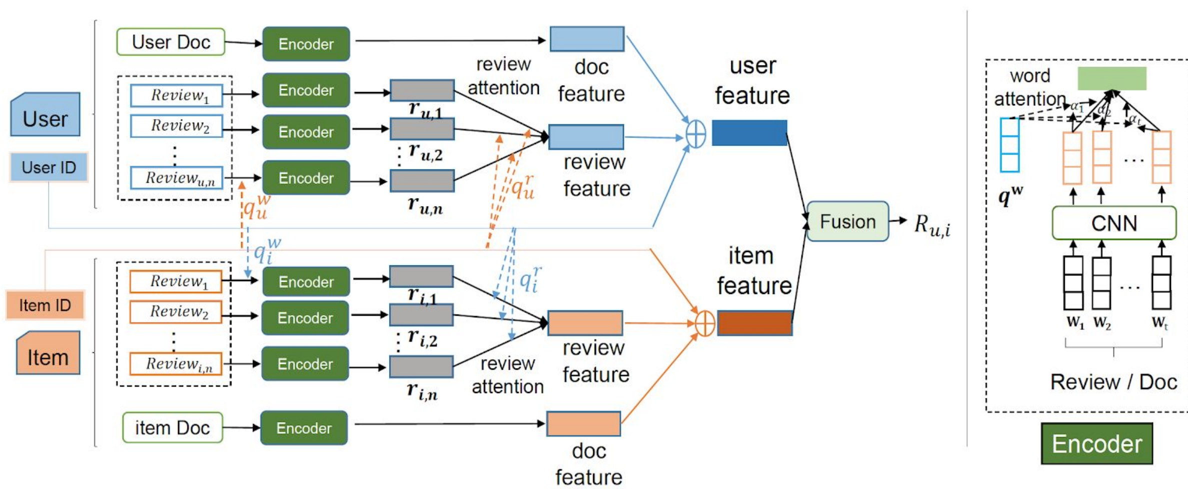


Figure 1: The framework of our *NRCA* approach.

- The framework above describes the model to be developed for rating predictions.

Document/Review encoder

1. Perform word embeddings where we represent a word a n-dimensional vector.
2. **Semantic Analysis:**
    a. We use CNN for extracting the semantic features of set of words(S).
    b. We generate a matrix C(say) of dimension(TxK) where T is the number of words in S and K is the number of filter for CNN layer, We have cm denoting the semantic feature for mth word of the review.
3. **Cross Attention :**
    a. As discussed above, since all words are not equally important for the user w.r.t different target items we assign weights to each word which define their relative usefulness. The attention weight is given by,

$$\alpha_m = \frac{\exp(g_m)}{\sum_{j=1}^{T} \exp(g_j)}, \; \alpha_m \in (0, 1) \; ,$$

$$g_m = \mathbf{W}(\tanh\,[\mathbf{q_w^u}; \mathbf{c_m}]) + b \; ,$$

    b. Then we obtain the representation vector of the *i*-th review of user *u* based on the item *i* via weighted summation of all words :

$$\mathbf{r_{u,i}} = \sum_{m=1}^{T} \alpha_m \mathbf{c_m} \; .$$

Review Level User/Item Encoder

---

1. Similar to word level attention as discussed above, different reviews contribute differently and users always have different preferences corresponding to different items.
2. Hence we use a similar cross attention mechanism to learn an item guidance representation of users.
3. We calculate weights of each review with respect to every particular user and normalise the weights between 0 and 1.

$$\beta_j = \frac{\exp(e_j)}{\sum_{k=1}^{n} \exp(e_k)}, \ \beta_j \in (0,1) \ ,$$
$$e_j = \mathbf{W_2}(\tanh\,[\mathbf{q_r^u}; \mathbf{r_{u,j}}]) + b \ ,$$

4. Then, we obtain the review-level feature p*u* of user *u* via aggregating all the reviews according to their weights:

$$\mathbf{p_u} = \sum_{j=1}^{n} \beta_j \mathbf{r_{u,j}} \ .$$

Fusion and Rating Prediction:

- We fuse the above three types of features for users and items to derive the final comprehensive representations

$$\mathbf{u} = \mathbf{d_u} \oplus \mathbf{p_u} \oplus \mathbf{u_{id}} \ ,$$

$$\mathbf{i} = \mathbf{d_i} \oplus \mathbf{p_i} \oplus \mathbf{i_{id}} \ ,$$

- Ratings : Finally we predict the rating:

$$R_{u,i} = \mathrm{W}_1^{\mathrm{T}}(\mathrm{u} + \mathrm{i}) + \mathrm{b_u} + \mathrm{b_i} + \mu \ ,$$

Experiments :

- Digital Music, Grocery and Gourmet Food, Video Games, Office Products and Tools Improvement selected from Amazon review

- Word embeddings : GloVe Model, dimension = 300

- The dimension of user or item ID embedding = 32

- Convolution : 100 convolution filters, window size = 3

- Dropout ratio = 0.7 and learning rate = 0.003

- Evaluation Metric : Mean Square Error(MSE)

Results:

- Dataset Used : Amazon fine food Reviews
- Ratings = 0 or 1(binary)
- Text preprocessing and train_test split
- Word Embeddings : Glove Model with output dimension=300
- Model:
    - Convolution layer with five different filter sizes with each of 200 filters

- Dense hidden layer with 128 hidden unit
- Regularisation Parameters : Dropout

- Output:
  - Model outputs an accuracy of 78.67% on the test dataset.
  - Example:
    - Sentence : “worst product ever”
    - Model Output : “Negative sentiment(Rating 0)with 52% accuracy

- Conclusion : The model doesn’t perform well on unseen data and overfits due to less training data and imbalanced training dataset.

References and Acknowledgements:

1. Datasets:
   a. https://www.kaggle.com/snap/amazon-fine-food-reviews
   b. https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

2. Research Paper - Neural Unified Review Recommendation with Cross Attention
   - https://dl.acm.org/doi/abs/10.1145/3397271.3401249

Thank You