# Predicting User Intent from Movie Reviews Using Deep Learning Methods

R Lokeshkumar

# Predicting User Intent from Movie Reviews Using Deep Learning Methods

**R Lokeshkumar, S Keerthi, Aditya Praveen Bora, K Jayakumar**

***Abstract*: Intent detection plays a vital role in understanding audience sentiment about particular movie. Predicting user in-tent on internet is a problematic task, which requires the combination of multiple sources of information, such as user profile, website activity or user's internet history. Deep learning plays an important role in text classification as well as in intent detection. Amongst them, most common methods used are recurrent neural networks (RNN) and convolutional neural networks (CNN). In this paper, we use these two deep learning methods to do intent detection on the Rotten Tomatoes dataset and try to tackle the problem of predicting the user intent, based on the reviews given to particular movie. We have compared these two methods based on the accuracy of intent detection as well as the time complexity and performance.***

***Index Terms*: Intent detection, neural networks, CNN, RNN.**

## I. INTRODUCTION

Movie reviews given by critics as well as audiences are important factors for giving ratings to a movie and predicting the success of movie. These reviews are usually in text for-mat in different languages and hence it is very difficult to accurately determine the user intent behind that review. The challenge of building an intelligent review system that will be able to correctly find intent requires a deep understanding of what are the intentions of various users with their reviews. Therefore, the main problem lies in defining this and then making educated guesses with precision. The best way to encounter this is to use some machine learning models. These models become more and more accurate with increase in data, and this data is collected directly from user reviews. Intents are the intentions of the reviewing user and what user thinks or means is known as intentions. We build this model as a coarse intent prediction, dividing the intents in two main categories, Good and Bad.

The first category is about reviews where the user thinks that the movie was good and has a positive opinion about that movie. The second category consists of reviews of the user where the user thinks that the movie was not good and overall has a negative opinion about that. It can be observed that these two classes split the reviews of the user into two main categories. This allows further modeling of the re-views system, such as including more sub-intents, originating from these coarse ones, which is described above. Natural Language Processing, or NLP for short, is defined as a sub-field of Artificial Intelligence (AI) which enables the automatic manipulation of natural language, like speech and text, by software. In a nutshell, it helps computers to under-stand and process human languages. The study of natural language processing has been around for more than 50 years and grew out of the field of linguistics with the rise of computers. One of the popular applications of Natural Language Processing (NLP) together with Machine Learning (ML) is "Text Classification", it's an application of Supervised Ma-chine Learning since it requires a labelled dataset for training a prediction model [7]. The basic goal of this is the classification the text data in dataset into one or more predefined categories. In our case, we use text classification for intent detection.

A general text classification pipeline has following main components: -

1. Training text: It is the input text or available data with labels through which our supervised learning model is able to learn and dissect the intent.
2. Feature Vector: Feature vector contains information of the different labels of the input data.
3. Labels: These are the classes of the input text data. We build our model such that it will predict the class or label of new data.
4. ML Algorithm: It is the algorithm which our model follows in order to deal with intent detection. In this case, we are considering CNN and RNN.
5. Predictive Model: A model which is trained on the input training dataset and is used to identify intent.

CNN: A convolutional neural network (CNN) is a class of deep, feed-forward artificial neural networks and it makes use of different multilayer perceptron's designed to require minimal preprocessing. These are inspired by animal visual cortex. CNNs are generally used in computer vision, however they've recently been applied to various NLP tasks.

RNN: A recurrent neural network (RNN) is a class of artificial neural network in which connections between nodes form a directed graph in a sequential manner. This allows it to display dynamic temporal behavior for a time sequence. Using the knowledge from an external embedding we can enhance the precision of your RNN because it integrates new information (lexical as well as semantic) about the words, an information that has been trained and distilled on a very large corpus of data.

**Revised Manuscript Received on April 07, 2019.**

**R Lokeshkumar**, Department of Computer Science, Vellore Institute of Technology, Vellore, India.
**S Keerthi**, Department of Computer Science, Vellore Institute of Technology, Vellore, India.
**Aditya Praveen Bora**, Department of Computer Science, Vellore Institute of Technology, Vellore, India.
**K Jayakumar**, Department of Computer Science, Vellore Institute of Technology, Vellore, India.

RNN is a sequence of neural network blocks that are linked to each other like a chain.

## II. . LITERATURE SURVEY

Two popular approaches are usually taken for prediction of intent, rule-based approach and statistical approach. In rule-based approach, for prediction of intent, we apply various rules at word as well query level. These rules are based on previous data. Rule-based approach don't have any statistical approach [8]. The main advantage of this approach is that there is that the predictions made are consistent as well as reliable and the predicted intent can be back-traced to basic rules. However, as database grow bigger continuously and as the user demand increases, these set of rules become increasingly complex, leading to reduced debugging possibility-ties as well as increased difficulty in modifying existing rule set. In statistical approach, we use different set of statistical approaches like machine learning algorithms and models. These models give better accuracy as the data increases and hence real-world data make Machine Learning models more accurate. The main drawback of this approach is that finding the most suitable set of parameters (trainable and not) is very difficult. Another drawback is, it is also difficult to construct a good model with best labels and creating a good evaluation metrics with good annotated data, so that the models can learn a distribution of data that will be accurate to newly acquired data. We usually choose the second approach, as it generally produces more accurate predictions. Some similar work has been done in detecting user intent from queries and reviews. In [1][7], they Implemented network feedback model for modeling of labels and extracted intent from the user query. They used these labels to train SVM (Support Vector Machine) to use on new data. In [2], they used both SVM and Naive Bayes algorithms to test similar features. They test the necessity of multi-label classification, talking about the obscurities of the requests and the limits of various single-label classification approach. In [3] they first label web pages and accordingly classify user queries in the three classes mainly based on the web pages classifier, and fuzzy rules based on user history and search engine history. In recent articles, the use of more advanced machine learning algorithms, such as LSTMs and word embedding's can be observed. In [4] they try to improve the off-the-shelf word embedding's using various language rules, such as keeping antonyms further away and keeping synonym's together, as close as possible, while keeping the original distances intact to the word neighbors. Then, they train a LSTM-based neural network model to predict user intent on two public datasets. Finally, in [5], they train a LSTM neural network model for both slot-filling and intent prediction, and show that this multi modal training improves the results of both tasks, on public datasets.

In this paper, we try to do similar to the ones described previously, using various word embedding's, the Glove pre-trained embedding [6] (with 100-dimension vectors). We use a Rotten Tomatoes movie dataset in which we annotate a small amount of the data, in order to have a correctly labeled dataset. Then, we train two main neural network models (CNN and RNN) and test efficiency of the modules, to validate the correctness of the two methods. We present the results of the majority of these combinations, using the ground truth labeled data for both multi-intent prediction and single-intent prediction.

## III. METHODOLOGY

*a)* Convolution Neural Network

Here CNN and RNN algorithms is being used for the dataset. The deep learning algorithms used identifies the ident and enables to classify the text in the dataset. Convolution Neural Network (CNN) is widely used in image classification problems. The major idea behind CNN is that background study of an image is sufficient. The main benefit is that having less parameters improves the learning time as well as reduces the training data amount. Instead of completely connected network of network of weights from every pixel, a CNN is sufficient to see at a small patch of the image. A convolution is the weighted sum of the pixel values of the image, as the sliding window slides across the entire image.

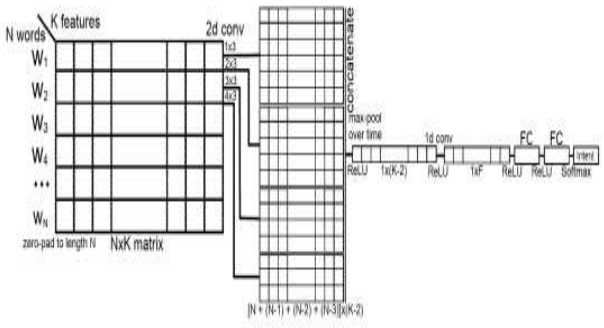


Fig.1 CNN architecture

The final model used here is a non-recurring model which uses the 2D convolutions to filter the multiple words in a stretch. Multiple independent convolutions are used having varying shapes which has the effect of filtering out words at the same time. For the proper understanding, a constant of 3 was kept, so all convolutions have a shape of i*3 where i represents the number of words which is being filtered at the same time. Mainly it is trying to combine features of multiple words in order to increase the window size of the query in order to capture longer context. All these features were combined in such a manner that it gives output to a larger matrix with features indexes corresponds to one word and the context sizes varying from 1 to n. The second step involves keeping the most important feature indexes of every features map which is equivalent to only keeping the most relevant feature of the most relevant words. The major operations are performed using max-pooling over time.The output received is passed through a 1D convolution, in order to decrease the size and then passed through a completely connected layer.

The fully connected layer produces three numbers which is similar to the recurrent models which were normalized to probabilities finally using a soft-max activation. Each of the convolution here is followed by a ReLU activation, which makes the model non-linear.

Here the data is being tokenized and made into sequences.

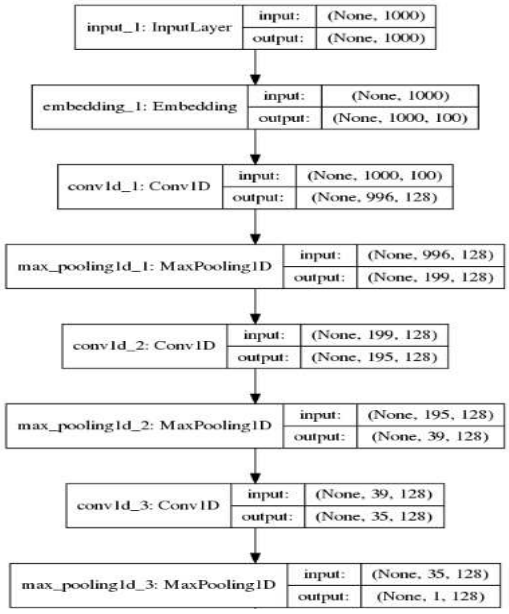


Fig.2 CNN layers

The number of unique tokens is being found out. The data is being filled out with the sequence words and the indices have been classified and arranged in such a way that the data is being classified in that manner. The data is then dived into training and testing data with the data sample and the label sample. The data have been divided into training and testing forms based on the number of validation samples. The final embedded matrix is being created and then the CNN is being applied to it.

```
tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)

word_index = tokenizer.word_index
print('Number of Unique Tokens',len(word_index))
```

```
Number of Unique Tokens 932
```

```
data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)

labels = to_categorical(np.asarray(labels))
print('Shape of Data Tensor:', data.shape)
print('Shape of Label Tensor:', labels.shape)

indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]
nb_validation_samples = int(VALIDATION_SPLIT * data.shape[0])

x_train = data[:-nb_validation_samples]
y_train = labels[:-nb_validation_samples]
x_val = data[-nb_validation_samples:]
y_val = labels[-nb_validation_samples:]
```

```
Shape of Data Tensor: (100, 1000)
Shape of Label Tensor: (100, 17)
```

Fig.3 tokenizing and training data

CNN consists of 3 major steps like sizing, convolution and pooling. After taking the embedded model the data is being reshaped in a manner that the sufficient data is being sized out from the entire data. To the data sized out convolution is being applied and the important data is being pooled out from the previous step. After all the data is being convolued and pooled the data is being concatenated and kept. The vec-tors is being flattened and dropped out and the labels have been predicted and normalized using the softmax activation function.

b) Recurrent Neural Network

In recurrent neural network, each movie review will be mapped into a real vector domain using a popular method known as word embedding. In this method the words are encoded as real-values vectors in a high dimensional space in which the similar words in terms of meaning will be translated to closeness in the vector space. Here keras will be used as a convenient way to transform the positive inte-ger representation into a word embedding by the embedding layer. Here each word will be mapped into a 32 length real valued vector. The total number of words that should be modelled is limited to 5000 most frequently used words and zero out of the rest of the words. And finally the sequence length in each review varies so each review is constrained to 500 words, pruning long reviews and pad the shorter re-views with zero values. After preparing the data LSTM mod-el is developed to classify the movie reviews. Here the data is being scrapped out using Beautiful Soup and is being made into an array of words.

```
# reading data
df = pd.read_excel('dataset.xlsx')
df = df.dropna()
df = df.reset_index(drop=True)
print('Shape of dataset ',df.shape)
print(df.columns)
print('No. of unique classes',len(set(df['class'])))
```

```
Shape of dataset  (100, 2)
Index(['message', 'class'], dtype='object')
No. of unique classes 17
```

```
macronum=sorted(set(df['class']))
macro_to_id = dict((note, number) for number, note in enumerate(macronum))

def fun(i):
    return macro_to_id[i]

df['class']=df['class'].apply(fun)
```

```
texts = []
labels = []


for idx in range(df.message.shape[0]):
    text = BeautifulSoup(df.message[idx])
    texts.append(clean_str(str(text.get_text().encode())))
```

Fig.5 reading and indexing data

The array of words taken from the dataset is being tokenized and is being checked out for the total number of unique to-kens. The sequences is being padded considering the maxi-mum length of the sequence. The data is then classified into labels and data. The indices are then arranged as per the la-bel. After arranging the labels, the validation samples is being splitted into training and testing values. The values have been splitted based on the validation samples. The x-train consists of the data sample up to the validation samples and the y-train consists of the label samples up to the validation samples.

Similarly the x-val consists of the data samples from the validation samples and the y-val consists of the label samples from the validation samples.

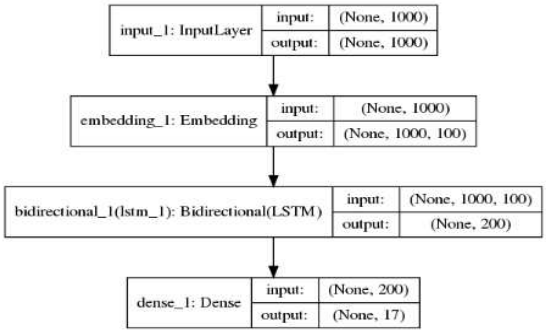


Fig.6 RNN layers

```
tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)

word_index = tokenizer.word_index

print('Number of Unique Tokens',len(word_index))
```

```
Number of Unique Tokens 932
```

```
data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)


labels = to_categorical(np.asarray(labels))
print('Shape of Data Tensor:', data.shape)
print('Shape of Label Tensor:', labels.shape)

indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]
nb_validation_samples = int(VALIDATION_SPLIT * data.shape[0])

x_train = data[:-nb_validation_samples]
y_train = labels[:-nb_validation_samples]
x_val = data[-nb_validation_samples:]
y_val = labels[-nb_validation_samples:]
```

```
Shape of Data Tensor: (100, 1000)
Shape of Label Tensor: (100, 17)
```

Fig.7 tokenizing and indexing in RNN

The words have been indexed and embedded and the length has been set to maximum sequence length and the data has been set to trainable form such that the data can be trained and tested and the algorithms is being applied to the embedded dataset. In order to make the data in a normalized exponential form it being given the soft-max function.

## IV. RESULTS AND DISCUSSION

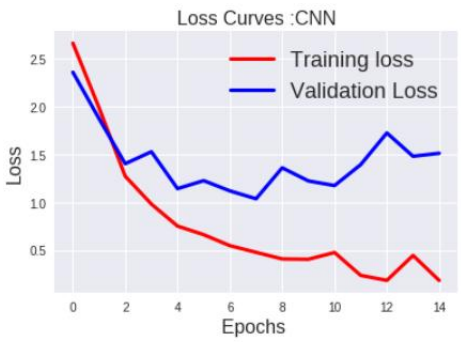


Fig.8 CNN loss curve

Here it can be observed that with increase in epoch there is an increase in validation loss. As for the training set it can see that with increase in epoch there is a decrease in the loss percent. A fluctuating changes in the loss of the validation data can be seen and as the epoch increases there is an increase in the loss of the data.

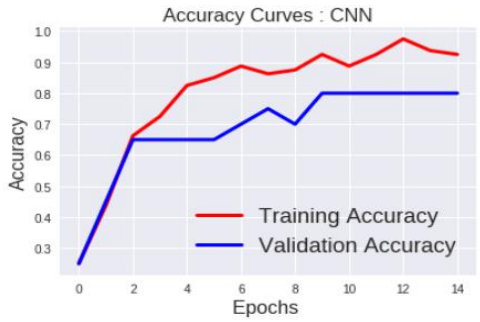


Fig.9 CNN Accuracy curve

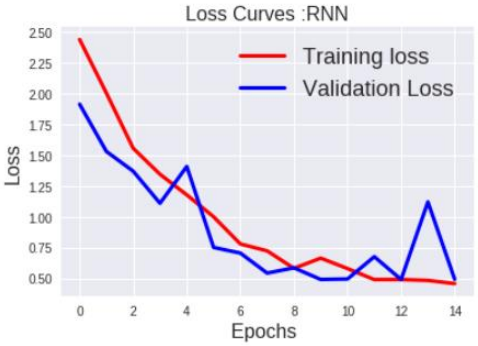


Fig.10 RNN Loss curve

As for the accuracy curves, it can be observed that for CNN the validation accuracy is lower than that of the training accuracy. And the validation accuracy becomes constant with the increase in the epochs. As it can be observed that the training data increases with the increase in the number of epochs. Here for CNN we have recorded the accuracy of 14 epochs.In the case of RNN it can be observed that the validation loss decreases with the increase in the number of epochs and there is a steep increase in the validation loss after a certain number of epochs. In an overall way it can be observed that the validation loss is less compared to the training data set loss for RNN.

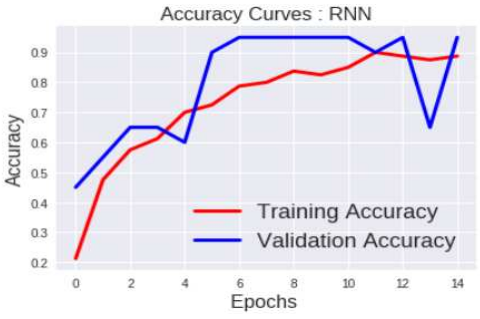


Fig.11 RNN Accuracy Curve

For RNN the accuracy of the validation data increases with epoch number till a certain limit. A gradual increase in the accuracy can be seen and after a certain period a dip can be observed for the testing data. There is a sharp increase in the accuracy after the dip and it gradually reaches to a constant stage and again continues the cycle with the epochs.

## V. CONCLUSION AND FUTURE SCOPE

It can be understood from the results that for the particular dataset RNN shows a better performance compared to CNN. The validation loss of RNN is much lesser compared to that of CNN. The accuracy is also observed higher for RNN com-pared to CNN. Even though RNN shows a sharp dip for some epochs the overall increase in accuracy is observed higher for RNN. As for CNN there is fluctuating accuracy range for the epoch values. Here the algorithm is used for only one dataset. The algorithms can be applied to numerous datasets and the accuracy and loss percent of the validation data can be rec-orded. A more precise accuracy, f1score and precision for more datasets can be recorded as a future work of this pro-ject. The values can be compared and even ensemble meth-ods using RNN, CNN and HAN can be proposed for this work. The deep learning methods can be combined to give the best accuracy for the datasets.

## REFERENCES

1. Dayong Wu, Yu Zhang, Shiqi Zhao, and Ting Liu. Identification of web query intent based on query text and web knowledge. In Pervasive Computing Signal Processing and Applications (PCSPA), 2010 First International Conference on, pages 128–131. IEEE, 2010.
2. Irazú Hernández, Parth Gupta, Paolo Rosso, and Martha Rocha. A simple model for classifying web queries by user intent.
3. Ariyam Das, Chittaranjan Mandal, and Chris Reade. Determining the user intent behind web search queries by learning from past user interactions with search results. In Proceedings of the 19th International Conference on Management of Data, pages 135–138. Computer Society of India, 2013.
4. R. Lokeshkumar & S.Durga "A Framework To Integrate Feature election Algorithm For Classification of High Dimensional Data", Volume. 4, Number.5, pp. 345-349, May 2016.
5. Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. Intent detection using semantically enriched word embeddings. In Spoken Language Technology Workshop (SLT), 2016 IEEE, pages 414–419. IEEE, 2016.
6. Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. arXiv preprint arXiv:1609.01454, 2016.
7. Lokeshkumar R, Maruthavani E, Bharathi A, "A New Perspective for Decision Makers to improve efficiency in Social Business Intelligence systems for Sustainable Development" Int. J. Environment and Sustainable Development, Inderscience Publishers, June 2018 , ISSN 1474-6778
8. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014.