Sentiment Analysis using Deep Learning

Subhashini Arunachalam
Computer Science Department
Drexel University
Philadelphia, PA 19104
USA
sa3643@drexel.edu

Madhu Anumula
Computer Science Department
Drexel University
Philadelphia, PA 19104
USA
mla332@drexel.edu

***Abstract* –** With the rapid growth of the online reviews and information conferred to us on daily basis, text classification has become crucial to maintain the information and to classify it. Such reviews on products, businesses, people can be helpful to customers as they can see any negative points about the products before they decide to purchase. In this paper, we have discussed a deep-learning method to leverage the existing opinionated content in order to offer better experience for customers to search about an entity. Basically, Long Short-Term Memory are those learning networks that work best when there is sequence of data, exploiting this feature we have implemented LSTM model as text classifier. We have also used Word2vec model to get a distinctive view to the data. Word2vec attempts to understand meaning and relationship among words, converts words and phrases into a vector or numerical format. Based on the idea that word2vec will bring extra semantic feature that aids in text classification, our work incorporates word2vec model to classify the reviews and to obtain the numerical statistic to indicate how important a word is to an opinion document of a particular entity.

Keywords: *word2vec; cosine similarity; Feedforward; recurrent neural networks; LSTM; supervised learning, semantic features.*

I. INTRODUCTION

Thrive in an era of social computing leads to huge expansion of online reviews on the web about products, businesses. If we take hotel domain as an example, many hotel management companies optimize the performance of their websites to improve the feedback from its users in order to retain the clients, or to offer better service to its clients. As a result of the effort taken by companies to forecast client behavior, there are multiple websites; blogs; social media; online forums, loaded with user opinions concerning the service received by the users. The massive amount of opinions by ordinary consumers can be useful for other people to make decisions regarding what product to purchase or what service to pick. It is a fact that 90% of consumers read online reviews before selecting a product and 88% of consumers trust online reviews like personal recommendations [3]. But in place of being an aid, the sumptuous amounts of reviews and multifaceted opinions make it difficult for the users to collaborate and evaluate. In our work we propose deep-learning techniques to analyze the client reviews, classify them that would in future become a base for future developments that would be helpful for users who want to search an entity.

II. BACKGROUND

Sentiment analysis is widely applied to extract and analyze the subjective information which are available in the websites, social media and many more sources. Two main approaches used for sentiment analysis are lexicon analysis which uses lexicons of pre-recorded sentiment and learning approach which is state-of-the-art but more computationally expensive deep learning to learn and generalize vector representation from words. In our work, we worked with deep learning approach to determine the polarity of the client reviews.

Text classification is widely seen as a supervised learning that is described as finding the categories of the new documents based on analyzing the training data that have labelled documents. As the amount of textual content of the new documents increases, it would become difficult to classify them. This is because the ability to identify the category of the new document depends on heavily upon the labelled training documents. In traditional information retrieval techniques such as bag of words model, a text is described as bag of its words by disregarding grammar and word order but keeping multiplicity [2].

A recent development called Word2vec model from google helps to represent a document in more intelligent way. In our work, we used word2vec model for added sematic features in order to help text classification. Word2vec takes text corpus as input and produce set of vectors as output, vectors of

similar words are grouped together in vector space (e.g., king - man + woman = queen). Similarity between two vectors of an inner product space is measured by cosine similarity. Here, words are clustered together and words are trained against other words that are closest to them in the input data [6].

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

In order to learn to recognize patterns in a set of data, neural networks are used. Neural networks generate a predicted output pattern when trained with the sample of the data and neural networks are algorithms that essentially applies series of computations to the matrices. Two traditional architectures of artificial neural networks (ANN) are feedforward neural network and recurrent neural network (RNN). Feedforward neural network accepts fixed sized inputs like binary numbers whereas RNN learns from the sequence of the data like text. Our work focuses on using long short term memory (LSTM) of RNN architecture to process the sequence of inputs. LSTM as the name suggest, is a network that does not just work on the current node but memorizes the previous nodes and computes the output based on the current as well as previous states, and because of this this feature we chose LSTM to classify the dataset.

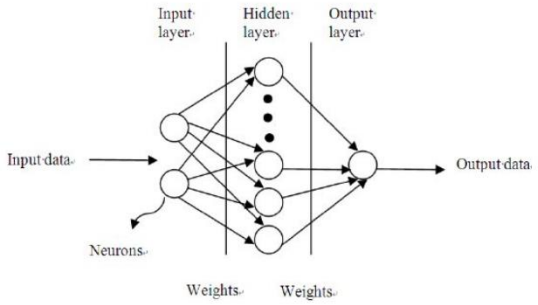

Fig: Neural model

III. APPROACH

Data Set − We built our model on Hotel reviews, and so fetched data corresponding to multiple Hotels from a country via online available resources that contains basic details like Hotel descriptive columns, user details, user experience in terms of reviews, ratings provided etc. We have split this data into Training, Validation and testing to build our model. Training data is the portion from where our model learns from and helps us fit our weights, validation data is part of the training process and helps prevent over fitting by letting us tune our hyper parameters accordingly. Testing data is used by the model to test itself by comparing its predictive labels to actual labels.

Data preprocessing − Before splitting the data into sets, we pre-processed our data using python NLTK library that allowed us to convert input text into individual words (tokenization), remove punctuations, delete words identified as stop words, remove any digits present.

The model we built needs the input data to be represented in terms of integers whose size need to be same for all inputs to ensure consistency in our input's dimensionality, so to this meet requirement, we used word2vec model of Google for vectorized representation of the reviews and padded it with zero at the end until it reaches the maximum possible sequence length which we set to 100. Below is a result of Word2vec representation of word 'neatness' when run on the corpus of available data.

```
In [299]: w2v_model['neatness']
__main__:1: DeprecationWarning: Call to deprecated `__getitem__` (Method will be
in 4.0.0, use self.wv.__getitem__() instead).
Out[299]:
array([-0.25304103,  0.13751885, -0.05397251, -0.17071488, -0.0210503 ,
       -0.02650489, -0.24118735, -0.01049802, -0.22588646, -0.05244441,
        0.04926011,  0.12845716,  0.02944663, -0.08196687, -0.02026809,
       -0.07228344, -0.06674962,  0.04096318,  0.17508687,  0.02386825,
       -0.00802732, -0.03859624, -0.04302422, -0.06147251, -0.0356821 ,
       -0.04670002,  0.01519069,  0.15517074,  0.0727896 , -0.04995331,
       -0.09097025, -0.14789176,  0.02295299,  0.09717714, -0.1511378 ,
        0.13033447, -0.04411807, -0.07665084, -0.24468991,  0.00994753,
        0.12059428, -0.05316051,  0.08957438, -0.09672701, -0.14090222,
        0.01081636, -0.02319094, -0.00061075,  0.03747756,  0.00580802,
       -0.04540211, -0.00098186,  0.15599991, -0.02913638, -0.0658805 ,
       -0.17895839, -0.03424169, -0.10090861,  0.0592809 , -0.12589252,
       -0.07878127, -0.01614173,  0.03559872, -0.07802472,  0.10485034,
       -0.01413729, -0.08146501,  0.06753872, -0.066369  ,  0.01208895,
       -0.0292157 ,  0.04943992,  0.04801199, -0.09583485, -0.00944928,
        0.11286517, -0.32028186,  0.01864311,  0.04817262,  0.05263095,
       -0.11400869,  0.08493919, -0.19625568, -0.14178222, -0.03997596,
       -0.01967227,  0.15033548, -0.0684198 ,  0.04673471,  0.10628846,
        0.09169002, -0.06714243,  0.03352977, -0.04512635, -0.04279153,
       -0.00193415, -0.02101138,  0.13053197, -0.23743977, -0.13519815,
       -0.1875134 ,  0.10399939, -0.07819241,  0.15093789,  0.00829504,
       -0.03208965, -0.06343087,  0.03209676,  0.10727949, -0.00204313,
        0.04782273,  0.02494976, -0.09560092,  0.0175411 ,  0.0146041 ,
        0.10701425,  0.18320294,  0.04409764,  0.13546363,  0.28021592,
        0.03174085,  0.03714693, -0.00723615, -0.02288887, -0.364527  ,
       -0.08710359,  0.01706932, -0.00499331,  0.02717666, -0.10613685,
       -0.06974352, -0.08581506, -0.05093613,  0.0613594 , -0.00300559,
        0.2025986 ,  0.2273705 ,  0.02183273,  0.26459464,  0.02781711,
       -0.14079742, -0.04485069,  0.11603566,  0.2995694 , -0.2400419 ,
        0.27948102,  0.11021092,  0.1248632 , -0.01895913, -0.11147392],
      dtype=float32)
```

Also, Word2vec is suitable to find similar meaning data from within the corpus, below are the sample results we achieved.

```
w1='comfortable'
model.wv.most_similar(positive=w1)
Out[211]:
[('comfy', 0.9141384363174438),
('spacious', 0.6747169494628906),
('super', 0.6701139807701111),
('nice', 0.6668509244918823),
('soft', 0.659796953201294),
('spotless', 0.6575110554695129),
('neat', 0.64920973777771),
('tidy', 0.6421093940734863),
('quiet', 0.63938307762146),
('tempurpedic', 0.6242756843566895)]
```

Fig: Similar words for the word 'comfortable'

Long Short-Term Memory Network –Recurrent Neural networks (RNN) are a class of neural networks whose connections between nodes resemble a directed cycle. RNN use internal memory or hidden state to process a sequence of input which makes it best suitable for sequence data processing. But RNN suffer from a drawback called vanishing gradient problem where the network loses the learning or the error rate as it traverses back in the network. LSTM model has been developed to overcome this issue with RNN. LSTMs have internal mechanisms called gates that help the network to regulate the flow of information.

Long Short-Term Memory network have the same working as a standard RNN, with a particular modification at the node structure level. Each node in RNN consists a input gate, output gate and hidden state, in LSTMs an additional gate called forget gate is added to the structure. These gates help in learning from the data and keeping only relevant knowledge and discarding the other. The cell state act as a framework or medium that transfers relative knowledge all down the sequence chain.[8]

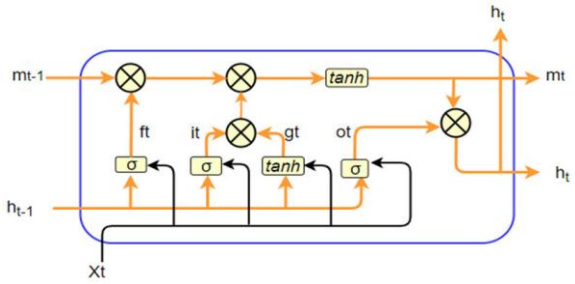

Fig: Node structure of LSTM model [8]

LSTM has two activation functions – Tanh function and Sigmoid function. These functions limit the values to required range regulating the output of the network. Tanh activation is used to regulate the values to be in the range of -1 to 1. And a sigmoid activation is similar Tanh but compresses values to be in range of 0 and 1. Using these activations the gates learn what data to keep and to discard based on their outputs. [8]

At time t, the LSTM will decide what data to keep and what data to lose based on the cell state. This decision is made by a sigmoid function, called Forget Gate. The gate takes knowledge from previous state as h_{t-1} and current input X_t and results in range 0 to 1, where 0 means value needs to be discarded and 1 mean retain.

$$f_t = \sigma(W^f x_t + U^f h_{t-1}) \quad [9]$$

Then the LSTM decides what new information from the input needs to be retained and discarded by first performing sigmoid function called Input gate on current input and previous hidden state. Along side the input and hidden state are passed through Tanh function and this output is multiplied with sigmoid output. This sigmoid output will help what information to keep from the tanh function. [8]

$$i_t = \sigma(W^i x_t + U^i h_{t-1}) \quad [9]$$

Once the previous hidden state and input are processed, the LSTM has enough data to update the cell state. First the cell gets point wise multiplied by the forget vector and all the values close to 0 are dropped. Then the output from the input gate is point wise added to update the cell state to new values. [8]

$$\tilde{C}_t = \tanh(W^n x_t + U^n h_{t-1})$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad [9]$$

Finally, the output gate decides what the next hidden state should contain that can be transferred to consequent nodes. To compute the same, first the previous hidden state and the input state are passed through sigmoid function. And we pass the updated cell state to tanh function and multiply these outputs to decide what data the hidden state should carry. [8]

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

$$h_t = o_t * \tanh(C_t) \quad [9]$$

These computed hidden state and cell state are passed over to the next layer/node.

Network building – We fed the pre-processed input data into the first layer of the neural network.

In the neural network, output from previous layer is given as input to the next layer. So, for the embedding layer of the neural network, we provided output from previous layer as input. Input dim (the size of the vocabulary in the text data): its value, based on how we coded the comments, is 5000. Output dim (the size of the vector space in which words will be embedded), this length is set to 300.

Output from embedding layer is fed to LSTM layer which is capable of learning and remembering entire sequences of data. In LSTM, dropout technique is used to prevent overfitting by randomly turning on and off the pathways in the network. Fully connected layer ensures every neuron in the previous layer is fully connected to every neuron in this layer. We used sigmoid activation function that takes in the vector of values and compress into a vector of output probabilities between 0 and 1 that sum to 1.

Input is fed into regression layer, which apply a regression operation to the input. Optimizer method in regression layer minimizes the given loss function as well as learning rate of the network. In our work, we worked with adam as optimizer to perform gradient descent and sparse categorical crossentropy as loss function to find the difference between the predicted and expected output. Once the network was built, we initialized and trained our model with the training and validation data set [5, 7].

IV. EVALUATION

We could achieve an accuracy of 88.9% using the LSTM network, below is the snippet of the code.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 200, 150)          4619400
_________________________________________________________________
lstm_5 (LSTM)                (None, 100)               100400
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 505
=================================================================
Total params: 4,720,305
Trainable params: 100,905
Non-trainable params: 4,619,400
_________________________________________________________________
None
Epoch 1/3
510581/510581 [==============================] - 39530s 77ms/step - loss: 0.7246 - accuracy: 0.8883
Epoch 2/3
510581/510581 [==============================] - 2766s 5ms/step - loss: 0.6951 - accuracy: 0.8838
Epoch 3/3
510581/510581 [==============================] - 3188s 6ms/step - loss: 0.6834 - accuracy: 0.8900
Accuracy: 88.94%n
```

We got lesser accuracy when the number of layers were decreased and the number of epochs were decreased. Below is the snippet of the result.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 200, 150)          4619400
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               100400
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 505
=================================================================
Total params: 4,720,305
Trainable params: 100,905
Non-trainable params: 4,619,400
_________________________________________________________________
None
Epoch 1/1
510581/510581 [==============================] - 3652s 7ms/step - loss: 0.7258 - accuracy: 0.7701
Accuracy: 77.29%
```

We assume that the efficiency of the algorithm can be further increased with modifications to the number of layers, training data set, trying out different activation functions etc., but LSTM takes huge computational space and time which makes it difficult to experiment under standard architectures and so we have concluded our project with limited results.

V. CONCLUSION AND FUTURE WORKS

In this paper we proposed a strong and efficient recurrent neural network called LSTM to classify texts. We have used Word2vec vector representation to normalize the data before fitting it to the network. The accuracy numbers seem reasonable and can be improvised using more corpus, and advanced processors that can handle the complexity of LSTM. As a work of future, we plan to model LSTM not just to classify the text but to classify and rank features from a text review. This type of analysis is called aspect-based sentiment analysis which can be easily achieved by LSTM with right training data and right parameters. Further LSTMs can also be improvised to for emotion analysis, multilingual sentiment analysis and further.

VII. BIBLIOGRAPHY

[1] Kavita Ganesan, ChengXiang Zhai: Opinion-based entity ranking, Information retrieval 15.2 (2012): 116-150. Available: http://kavita-ganesan.com/wp-content/uploads/ 2010/03/EntityRank.pdf

[2] Joseph Lilleberg, Yun Zhu, Yanqing Zhang: Support Vector Machine and Word2vec for Text Classification with Semantic Features, 2015. Available: http://www.cs.unibo.it/~danilo.montesi/CBD/Articoli/2015_Support%20vector%20machines%20and%20Word2vec%20for%20text%20classification%20with%20semantic%20features.pdf

[3] “customer review” [online] Available: https://en.wikipedia.org/wiki/Customer_review

[4] C.A. Martin, J. M. Torres, R. M. Aguilar, and S. Diaz: Using Deep Learning to Predict Sentiments: Case Study in Tourism, 23 September 2018. Available: https://www.hindawi.co /journals/complexity/2018/7408431/#B14

[5] Jin Wang, Liang-Chih Yu, K. Robert Lai and Xuejie Zhang: Dimensional Sentiment Analysis using a Regional CNN-LSTM Model, 7 August 2016. Available: https://www.aclweb.org/anthology/P16-2037.pdf

[6] “Word2vec” [online] Available: https://skymind.ai/wiki/word2vec

[7] Joana Souza, Alcione Paiva Oliveira, Guidson Coelho de Andrade, Moreira Alexandra: A Deep Learning Approach for Sentiment Analysis Applied to Hotel’s Reviews, December 2017. Available: https://www.researchgate.net/publication/325268267_A_Deep_Learning_Approach_for_Sentiment_Analysis_Applied_to_Hotel%27s_Reviews.

[8] Illustrated guide to LSTMs and GRUs a step by step explanation Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[9] Deep Learning for Sentiment Analysis: A Survey https://arxiv.org/ftp/arxiv/papers/1801/1801.07883.pdf