Text Data Classification Using Deep Learning Long Short-Term Memory Network

Amandeep Kaur[1], Abhinash Singla[2]
[1]Department of CSE, BGIET, Sangrur
[2]Department of HOD CSE, BGIET, Sangrur

Abstract- *Studies show that a Long Short-Term Memory (LSTM) Recurrent Neural Network can be used for text classification. The strength of the LSTM is that it can capture information in any part of the document. It also allows the model to account for the specific order of words with higher accuracy. In this paper text data classification using deep learning long short-term memory network has been presented*

Keywords- Long Short-Term Memory (LSTM), Text Classification, Deep Learning, Neural Network

I. INTRODUCTION

To label different types of text documents is both important and desirable. There are plenty of different situations where this is useful. Automating these tasks is therefore something of great value if the automated system performs on a par with, or better than, humans. Labelling essays with grades is an example of a task that is time consuming but also important, which is why a lot of research has been put into Automated Essay Scoring [1]. Removing posts from social media platforms which are against the terms of use or illegal such as hate speech or threats of physical violence is another case where automation, if done right, would be beneficial. Automated labelling of texts can also be useful in other cases. Classifying e-mails as spam, classifying reviews as either positive or negative and assigning topics to Wikipedia articles [2] are more examples of useful applications. When assigning topics to Wikipedia articles, the number of target classes is larger. The literature makes a distinction between the case when the target classes are binary and when there are several possible target classes, where the latter problem is more complex. A different text classification task is when a document can be labelled with several labels. This is referred to as a multi-label classification task. If the label-space is very big, the task becomes an extreme multi-label classification task (XMTC). An example of such a problem is the challenge proposed by [3]. The objective of this task was to assign several Medical Subject Headings, also known as MeSH, to new PubMed large database of medical articles documents. Text classification tasks can be summarized as four different types: one out of two classes, one out of multiple classes, several labels out of a limited number of labels and several labels out of extremely many labels [4]. Previously, there has not been any papers which deal with more than about 50 classes in a deep learning framework. The data used in this thesis has over 1000 target classes which makes this type of classification problem uncharted territory. The objective of the thesis is to investigate the performance of a neural network transfer learning technique, known as ULMFiT [5], on a task similar to the second type; to determine which one class out of multiple classes that a text belongs to. This can be thought of as a fifth category of text classification task; a highly multiclass classification task. In the next section there will be an overview of related work. Then, the data and method will be introduced. A benchmark classifier will be constructed for comparison purposes. Lastly, an implementation of ULMFiT on a highly multiclass classification task was presented along with results and a conclusion.

II. TEXT CLASSIFICATION TECHNIQUES

Various text classification techniques were initially identified through Wikipedia and other encyclopedias and corroborated with the content of various research articles. The major approaches were further arranged as a tree structure after analyzing the similarities and differences among these various approaches along with their respective algorithms. Generally, a classification technique could be divided into statistical and machine learning (ML) approaches. Statistical techniques purely satisfy the proclaimed hypotheses manually, therefore the need for algorithms is little, but ML techniques were specially invented for automation [6].

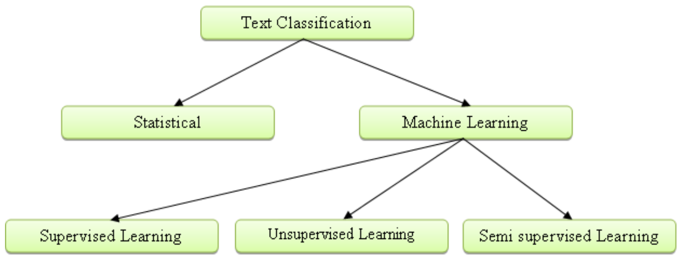

Fig 1. Shows text classification techniques

2.1 Statistical Approach

Statistical techniques are purely mathematical processes, and they act as the mathematical foundation for all other text classifiers. It works similar to a computer program, executing the given instructions without any ability of its own.

2.2 Machine Learning Approach

The increase in data volume, velocity, and variety called for automation in text processing techniques including text classification. In some situations, defining a set of logical rules using knowledge-engineering techniques and based on expert opinions to classify documents helps to automate the classification task. Text classification could be divided into three categories: supervised text classifica-tion, unsupervised text classification, and semi-supervised text classification based on the learning principle followed by the data model [7]. In machine learning terminology, the classification problem comes under the supervised learning principle, where the system is trained and tested on the knowledge about classes before the actual classification process. Unsupervised learning occurs when labeled data is not accessible. The process is complicated and has performance issues.

2.1.1Supervised learning

Supervised learning is the most expensive and highly difficult of the three. The main reason behind this notion is that it requires a human intervention while assigning labels to classes which is not pos-sible in large datasets. Though the work flow mimics the techniques followed in AI processes, it is time consuming. It is also called inductive learning in ML [8]. Supervised learning becomes expensive when different data distributions, different outputs and different feature spaces occur as in heterogeneous text corpora. One of the most widely used supervised methods is maximum likelihood estimation

2.1.**2 Unsupervised learning**

Unsupervised learning is a type of ML algorithm where, inferences are drawn from the data by clus-tering data into different clusters without labeled responses i.e. expected outcomes. In other words, no training data is provided to the system. It appears complex initially, but when more data is fed into the model, the algorithm refines itself to efficiency. Principal component analysis, clustering and self-organizing maps are frequently used in unsupervised learning. In many scenarios clustering is the same as unsupervised learning. Many times, expert knowledge required to label the samples is either non-existent or inadequate. In such case, self-organizing maps and correlation coefficient are used to cluster the documents and use it to label the documents for further classification [9]. It eliminates the curse of dimensionality and expert intervention as well. This kind of hybrid model is more suitable for high volume data.

2.1.3 Semi Supervised Learning

Semi-supervised learning is a combination of supervised and unsupervised learning techniques. This type of learning employs small amount of labeled data and large amount of unlabeled data for train-ing. The labels are assigned by combining labeled and unlabeled instances, as unlabeled data mitigate the effect of insufficient labeled data on classifier accuracy. Some of the SSL techniques are such as self-training or self-teaching or bootstrapping, co-training, transductive SVMs, generative models and graph-based methods. Vector space models are mostly used in language processing problems to address natural language semantics that supposes words in similar contexts have similar meanings. Meaning values are calcu-lated according to the Helmholtz principle. This model is non-iterative but effective in augmenting the efficiency of classifier. The system can be combined with semantic kernels that smooth docu-ment term vectors using term to term semantic relations.

III. NEURAL NETWORKS IN MACHINE LEARNING

Artificial neural networks (ANNs) work in the same way as human brain in arriving at a decision. Swarm intelligence and evolution algorithm are used to generalize a neural network model. It works on the virtue of learning and evolution with minimal or no human intervention. For data classification, competitive co-evolution algorithm based neural network model is suggested. Radial Basis Function is the ANN component as it employs faster learning algorithms. It has a compact network architecture that increases classification accuracy. Also, evolutionary algorithms have a tendency to perform well in dynamic environments by learning rules on the fly and highly adaptive of 'fuzzy' characteristics. Neural networks are also popular among cases where a hierarchical multi-label classification approach is required. A popular way to approach different tasks in NLP is to use a Long Short-Term Memory Recurrent Neural Network. The strength of the LSTM is that it can capture information in any part of the document.A simplified structure of the two-layer, feed-forward network can be seen in Fig. 2 It takes $x = \left[x_1 x_2 \mathrm{K}\ x_p \right]^T$ as input vector and produces $z = \left[z_1 z_2 \mathrm{K}\ z_k \right]^T$ as output vector where p is the number of variables and K the number of classes. In this network, the input x is transformed into z through one layer of hidden units and activation functions.

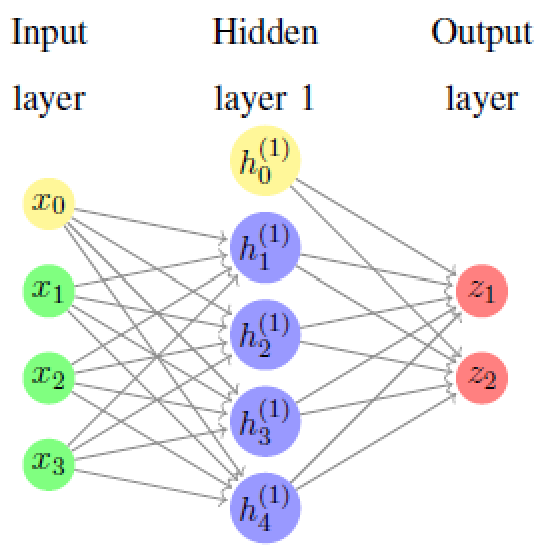

Figure 2 A feed-forward neural network with three input variables, four hidden units and two output variables.

The intercept is represented by $x_0 = h_0^{(1)} = 1$. The arrows between the nodes are the weights and there are also activation functions between the layers but they are not visible in this simplified illustration. In the following equations, the Rectified Linear Unit (Nair and Hinton 2010) activation function is used. It is defined as $\sigma(x) = \max(0, x)$ and introduces non-linearity into the network. The hidden units are then defined as

$$h_i = \sigma\left(\beta_{i0}^{(1)} + \beta_{i1}^{(1)} x_1 + \beta_{i2}^{(1)} x_2 \mathrm{K}\ \beta_{ip}^{(1)} x_p\right) \quad i = 1, 2, \mathrm{K}\ , M. \tag{1}$$

which is a linear function of the input variables put through the ReLU activation function. M . is the number of hidden units in this layer. They can also be written in matrix notation

$$h - \sigma\left(b^{(1)} + \beta^{(1)} x\right) \tag{2}$$

where $h = [h_1 h_2 \mathrm{K}\ h_M]^T$ is the vector of hidden units and

$$\beta^{(1)} = \begin{bmatrix} \beta_{11}^{(1)} & \beta_{12}^{(1)} & \mathrm{K} & \beta_{1p}^{(1)} \\ \beta_{21}^{(1)} & \beta_{22}^{(1)} & \mathrm{K} & \beta_{2p}^{(1)} \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ \beta_{M1}^{(1)} & \beta_{M2}^{(1)} & \mathrm{K} & \beta_{Mp}^{(1)} \end{bmatrix} \quad b^{(1)} = \begin{bmatrix} \beta_{10}^{(1)} \\ \beta_{20}^{(1)} \\ \mathrm{M} \\ \beta_{M0}^{(1)} \end{bmatrix}$$

is the weight matrix and vector of intercepts used to transform the input into hidden units. The superscript refers to that this weight matrix and intercept vector corresponds to the first layer in the network. In the two-layer example, the output is expressed as a function of the hidden units as

$$z_i = \beta_{i0}^{(2)} + \beta_{i1}^{(2)} h_1 + \beta_{i2}^{(2)} h_2 \mathrm{K}\ \beta_{iM}^{(2)} h_M, \quad i = 1, 2, \mathrm{K}\ , K. \tag{3}$$

In matrix notation

$$z = b^{(2)} + \beta^{(2)} h \tag{4}$$

Where

$$\beta^{(2)} = \begin{bmatrix} \beta_{11}^{(2)} & \beta_{12}^{(2)} & \mathrm{K} & \beta_{1p}^{(2)} \\ \beta_{21}^{(2)} & \beta_{22}^{(2)} & \mathrm{K} & \beta_{2p}^{(2)} \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ \beta_{M1}^{(2)} & \beta_{M2}^{(2)} & \mathrm{K} & \beta_{Mp}^{(2)} \end{bmatrix} \quad b^{(1)} = \begin{bmatrix} \beta_{10}^{(2)} \\ \beta_{20}^{(2)} \\ \mathrm{M} \\ \beta_{M0}^{(2)} \end{bmatrix}$$

and M and K are the number of hidden units and output classes respectively. To extend this two-layer network into a deep neural network with L layers it can be represented in matrix notation as

$$\begin{aligned} h^{(1)} &= \sigma\left(b^{(1)} + \beta^{(1)} x\right) \\ h^{(2)} &= \sigma\left(b^{(2)} + \beta^{(2)} h^{(1)}\right) \\ &\mathrm{M} \\ h^{(L-1)} &= \sigma\left(b^{(L-1)} + \beta^{(L-1)} h^{(L-2)}\right) \\ z &= b^{(L)} + \beta^{(L)} h^{(L-1)} \end{aligned} \tag{5}$$

Note that the ReLU is not used when generating z, instead, if the network is trained for classification, z is put through a softmax activation to convert the values into probabilities. The softmax is defined as

$$soft\max(x) = \frac{1}{\sum_{j=1}^{K} e^{z_j}} \left[e^{z_1} e^{z_2} \text{K } e^{z_K} \right]^T$$

(6)

The softmax function will assign values close to one for the largest z_i and close to zero for all others, unless the largest and second largest are very close.

3.1Training a Neural Network

When training a machine learning model one wants to find the parameter values which minimize a loss function. Given the model

$$y = X\beta + \in$$

(3.1)

a closed form solution that minimizes the MSE loss function can be directly calculated as

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T y$$

(3.2)

If the amount of parameters is extremely large, one could instead use an algorithm called Gradient Descent. The gradient descent method, when applied to linear regression, tries to minimize an appropriate loss function, e.g. MSE. To make the partial derivatives look nicer we define a slightly modified MSE as

$$L\left(\hat{\beta}_0, \hat{\beta}_1, \text{K } \hat{\beta}_p\right) = \frac{1}{2n} \sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

(3.3)

Gradient descent then takes the partial derivatives in each iteration with regard to all $\hat{\beta}_j$

$$\frac{\partial L\left(\hat{\beta}_0, \hat{\beta}_1, \text{K } \hat{\beta}_p\right)}{\partial \hat{\beta}_j} = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)}\right) x_j^{(i)}, \quad j = 1, 2\text{K } p.$$

(3.4)

Then it will update all parameters simultaneously with learning rate $\gamma > 0$ such that

$$\hat{\beta}_j^{(new)} = \hat{\beta}_j^{(old)} - \gamma \frac{1}{n} \sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)}\right) x_j^{(i)}, \quad j = 1, 2\text{K } p.$$

(3.5)

This update scheme is repeated until the difference between loss functions between iterations is sufficiently small. Then we can say that the loss function, which in this case is convex, has converged to its global minimum Deep neural networks often have millions of parameters and sometimes billions. The gradient can therefore not be calculated for all parameters and observations every time. Luckily, gradients between subsets of the data are often similar [24]. Therefore, it is possible to split the dataset into mini-batches and then calculate the gradient on each mini-batch.

The size of the mini-batches is determined by the memory in the computers GPU and is often somewhere between 32 and 256 [25]. The calculation of gradients is done with back-propagation [26]. Optimizing the parameter values through calculating gradients on mini-batches is known as Stochastic Gradient Descent (SGD). The parameters are updated in a similar manner to Equation 3.5 but n is replaced by the mini-batch size. When applying SGD with an adaptive learning rate scheme and adaptive momentum, the optimization algorithm must be able to handle this. The Adam optimizer [27] is a common choice under these circumstances since it is fast and can handle the adaptive learning rate and momentum scheme. Therefore, the Adam optimizer will be used in this thesis. When training a neural network for classification, the MSE loss function, described in Equation 3.5, is replaced by another loss function known as cross-entropy loss. It is defined as

$$L(x_i, y_i, \theta) = -\sum_{k=1}^{K} y_{ik} \log\left(p\left(k \middle| x_i; \theta\right)\right) = -y_i^T \log\left(soft\max\left(z_i\right)\right)$$

(3.6)

With xi being the predictors of observation i, yi being a one-hot encoded vector where the correct label of observation i is coded as 1 and the rest are 0 and _ are the current parameters of the model. It reduces to the negative logarithm of the probability assigned to the correct class by the softmax function. Thus, it penalizes the model for assigning high probabilities to incorrect classes. Correct guesses, especially when assigned probabilities close to 1, will yield a low loss. The task of optimizing the classifier can then be described in equation form as

$$\hat{\theta} = \arg_\theta \quad \min \frac{1}{n} \sum_{i=1}^{n} L(x_i, y_i, \theta)$$

(3.7)

A text can be considered as a sequence of words which might have dependencies between them. A long short term memory (LSTM) network is a type of recurrent neural network (RNN) that have long term dependencies between time steps of sequential data. In this work, LSTM neural network has been used to learn and use long term dependencies fro classification of text data. To input is a text file applied to an LSTM network. First the text data has been converted into numeric sequences. This has been achieved by using a word encoding which maps documents to sequences of numeric indices. To achieve better results a word embedding layer has been included in the network. Word embeddings has been used to map words in a vocabulary to numeric vectors rather than scalar indices. These embeddings have captured semantic details of the words and will result in generation of words which have similar meanings to have similar vectors. This has also helped into modeling relationships between words through vector arithmetic. The simulation work has been classified into four major steps:

a) Import and preprocess the data.
b) Convert the words to numeric sequences using a word encoding.
c) Create and train an LSTM network with a word embedding layer.
d) Classify new text data using the trained LSTM network.

Table 4.1: Factory Data in Tabular Form after Labeling and Categorization

Items are occasionally getting stuck in the scanner tools	Mechanical Failure	Medium	Readjust Machine	45
Loud rattling and banging sounds are coming from assembler pistons	Mechanical Failure	Medium	Readjust Machine	35
There are cuts to the power when starting the plant	Electronic Failure	High	Full Replacement	16200
Fried capacitors in the assembler	Electronic Failure	High	Replace Components	352
Mixer tripped the fuses	Electronic Failure	Low	Add to Watch list	55
Burst pipe in the constructing agent is spraying coolant	Leak	High	Replace Components	371
A fuse is blown in the mixer.	Electronic Failure	Low	Replace Components	441
Things continue to tumble off of the belt	Mechanical Failure	Low	Readjust Machine	38

IV. RESULTS AND DISCUSSIONS

The events in the data has been classified by the labeling it. Then the data is partitioned into a training partition and has been held-out the partition for validation and testing. The holdout percentage has been taken as 10%. To check that the data has been imported correctly, the training text data has been visualized using a word cloud.

The data has been tokenized and preprocessed using the following three steps:

- Tokenize the text using “tokenized Document”.
- Convert the text to lowercase using “lower”.
- Erase the punctuation using “erase Punctuation”.

To input the documents into an LSTM network, word encoding has been used to convert the documents into sequences of numeric indices. Then the documents have been padded and truncate to make them of same length. To pad and truncate the documents, first a target length has been chossen, and then truncate the documents that are longer than it and left-pad documents that are shorter than it. For best results, the target length should be short without discarding large amounts of data. To find a suitable target length, a histogram of the training document lengths shown in Figure 4.3 is simulated. As most of the training documents have fewer than 11 tokens. So, we have used 11 as the target length for truncation and padding.

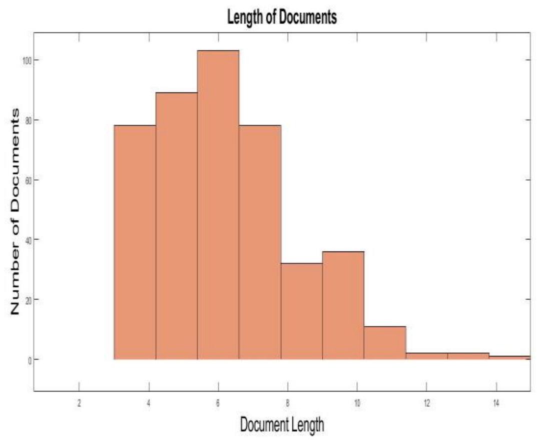

Figure 4.3: Histogram to Decide for Target Length

The final and most important step is to develop the LSTM network architecture. To input sequence data into the network we have used an input layer size of 1. The size of second layer of neurons has been taken as 70. This layer is a word embedding layer and has the same number of words as the word encoding. Then the number of hidden neurons has been set as 100. Lastly, to use the LSTM layer for a sequence to label classification problem the output mode has been set as “last”.

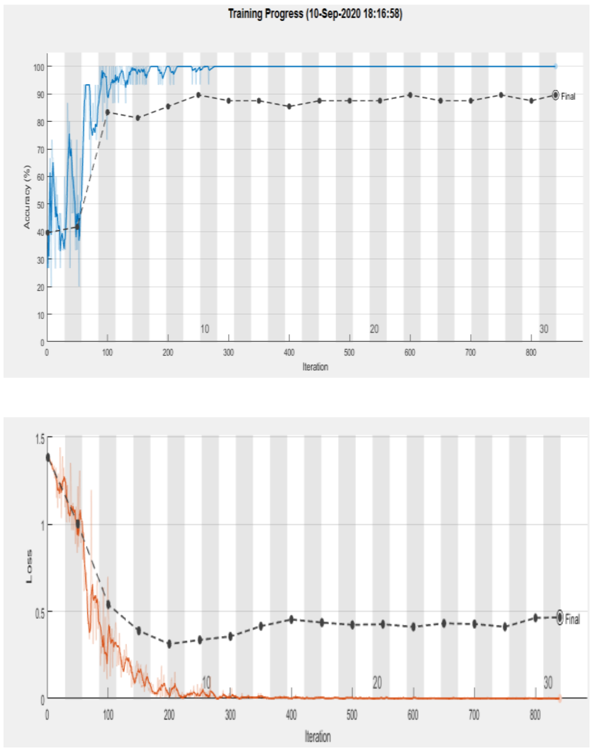

Figure 4.4: LSTM Network Training

To predict and testing the accuracy of the trained LSTM network, a test data has been used with the event type classified into three new reports. The string array containing the new reports has been created as shown:

"Coolant is pooling underneath sorter."
"Sorter blows fuses at start up."
"There are some very loud rattling sounds coming from the assembler."

The text data has been preprocessed using the preprocessing steps as done during training the documents. The text data has been converted into sequences with the same options as done during the training sequences process. Then the new sequences have been classified using the trained LSTM network. The following classifications have been obtained from the testing data:

- Leak
- Electronic Failure
- Mechanical Failure

From the classification results it has been concluded that LSTM network has classified the tesing data with 100% accuracy.

V. CONCLUSION

In the presented work the technique classify text using LSTM with variant number of words that have been grouped to be used as input features. LSTM techniques improves gesture recognition accuracy and minimizes the false-positive rate as well as the time complexity

REFRENCES

[1] J. Wang et al., “Spatiotemporal Modeling and Prediction in Cellular Networks: A Big Data Enabled Deep Learning Approach,” Proc. 36th IEEE INFOCOM, Atlanta, GA, USA, May 2017, pp. 1–9.
[2] A. Alahi et al., “Social LSTM: Human Trajectory Prediction in Crowded Spaces,” Proc. 29th CVPR, Las Vegas, NV, USA, June 2016, pp. 961–71.
[3] N. I. Sapankevych and R. Sankar, “Time Series Prediction Using Support Vector Machines: A Survey,” IEEE Comput. Intell. Mag., vol. 4, no. 2, May 2009, pp. 24–38.
[4] W.-C. Hong, “Application of Seasonal SVR with Chaotic Immune Algorithm in Traffic Flow Forecasting,” Neural Computing and Applications, vol. 21, no. 3, Apr. 2012, pp. 583–93.
[5] Frolova D, Stern H, Berman S (2013) Most probable longest common subsequence for recognition of gesture character input. IEEE Trans Cybern 43(3):871–880.
[6] Gao L, Bourke A, Nelson J (2014) Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems. Med Eng Phys 36(6):779–785
[7] Goyal M, Shahi B, Prema K, Reddy NS (2017) Performance analysis of human gesture recognition techniques. In: 2017 2nd IEEE international conference on recent trends in electronics, information and communication technology (RTEICT), IEEE, pp 111–115