ISSN: 0976-3376

Available Online at http://www.journalajst.com

ASIAN JOURNAL OF SCIENCE AND TECHNOLOGY

Asian Journal of Science and Technology
Vol. 14, Issue, 06, pp. 12555-12560, June, 2023

RESEARCH ARTICLE

LABELING OF TEXT DATA USING AUTOENCODERS

***Praveen Thenraj Gunasekaran, Selvakuberan Karuppasamy and Subhashini Lakshminarayanan**

Artificial Intelligence, Accenture, Chennai, India

ARTICLE INFO

Article History:
Received 03rd March, 2023
Received in revised form
26th April, 2023
Accepted 14th May, 2023
Published online 20th June, 2023

Keywords:
Unlabeled text data, Auto Labeling, AutoEncoders, Clustering.

ABSTRACT

Machine learning has come a long way in solving business use cases that has remained a nightmare to human. Today machines learn data in ways like human, machine learning has matured so much that all it requires is data and it can solve any problem if the correct data is provided. Among the different learning techniques, we have in current ML world, supervised learning is a popular technique where the model learns from labeled dataset. The model tries to learn the pattern from the data and tries to correlate the independent and the dependent variable. But the challenge in real time is we don't have the readily available labeled data which applies to unstructured text as well. Given the volume of the text data available and the multiple sources available, it would take humongous efforts to label these text data manually. This has led to the rise of many unsupervised techniques to learn the data for solving use cases. However, in-spite of numerous improvements in the domain of unsupervised learning, the supervised learning continues to one of the preferred techniques for humans to train machines. The objective of this paper is to use AutoEncoders combined with clustering technique to label the unlabeled text training data when the number of classes for the dataset is known.

Citation: ***Praveen Thenraj Gunasekaran, Selvakuberan Karuppasamy and Subhashini Lakshminarayanan.*** **2023.** "Labeling of text data using autoencoders", *Asian Journal of Science and Technology*, 14, (06), 12555-12560.

INTRODUCTION

In the era of deep learning, labelling the training data manually is a very tedious task, given the volume of training data that is being used. With the advent of machine learning which can solve many use cases in many domains, there must be techniques to solve its own problem of getting the data labeled for training purposes. There are also readily available labeling tools which can help labeling unlabeled dataset but still the reliability of these packages remains a question when it comes to critical business scenarios. In this paper, we propose a simple solution, based on autoencoder and clustering to solve the problem of labeling unlabeled text data. The solution consists of four parts 1. Embed the training dataset 2. Extract important features of the training dataset 3. Clustering of the lower dimension representation 4. Keyword identification from each cluster.

BACKGROUND

Text data labeling: Text data is a form of unstructured data. There are various sources of text data especially with the advent of internet and social media, the volume of unstructured data available also has increased linearly. Increase in volume also means annotating this huge volume of text data involves huge amount of human effort. Since we are dealing with big data, human intervention for such a huge volume of data would result in more resource necessity, accuracy in annotation as different people with different perceptions would be involved and also increased cost. In the process of continuous improvement, there has been some cool techniques semi-supervised learning that has been identified to solve the problem of unlabeled dataset. Unsupervised techniques also can be used to label the training data whereas semi-supervised techniques make use of a considerable portion of the training data that has already been labeled and uses them to learn and label the remaining dataset.

LITERATURE REVIEW

In [1], the authors have used an autoencoder and clustering based technique to solve the problem of labeling image dataset. The authors have used MNIST dataset for this experiment. [7] A Siamese network-based architecture to derive the sentence embeddings of a given pair of sentences. This approach is a modified version of the pretrained BERT model, and it generates more relevant embeddings with much reduction in computation time as well. [4]uses Deep Autoencoders along with SVM as a classification layer for classifying the images. The authors have used MNIST dataset for this work and have obtained 99.8% accuracy. [8] This paper marked a new era in the domain of NLP. The authors realized the need for understanding the contextuality of the tokens in a sentence and came up with two architectures namely CBOW and Skipgram to generate word embeddings for English language that can be used across any tasks.
[6] The authors in the paper have used K-Means algorithm as clustering technique for clustering the similar national anthems of different countries of the world. The authors have used TF-IDF as mechanism to extract the features from the documents and then used K-Means algorithm to cluster the documents. In [2], the authors have

used an autoencoder and clustering based architecture to identify the optimal number of clusters from the unlabeled text dataset. The authors have used Barez dataset from which the embeddings are created using pretrained model ParseBERT. In this paper, the authors have used Silhouette score to evaluate the clusters and find the optimal number of clusters. [3] analyses the various forms of autoencoders. The authors have discussed about the following forms of autoencoders like sparse, denoising, contractive, variational, disentangled autoencoders. The authors have also discussed about the various applications of autoencoders like classification, clustering, generative, anomaly detection, recommendation, dimensionality reduction. [5]. In this paper, the authors have used an improved version of Denoising Autoencoders for extracting the important features and then added a softmax layer as classification layer. It was observed that the improved version of the Denoising autoencoders performed better than normal denoising auto encoder and a plain KNN classifier. The accuracy of the denoising autoencoder stood at 95%. [9] The authors have proposed a sub word based embeddings in this approach to overcome the shortcoming of out of vocabulary tokens in case of generating embeddings. Also generating sub word level embeddings proved to be efficient when handling domain specific vocabulary and misspelt tokens.

Architecture: Our architecture consists of three modules namely feature extraction, clustering and keywords identification module that combine to achieve the concept of labelling a text dataset.

Auto Encoders: An Autoencoder architecture consists of two neural network modules which includes encoder and decoder. The encoder module can be considered as a simple compression module that compresses the input data to a lower dimension while trying to retain the important features. The layer which represents the input in the lowest dimension in this architecture is called bottleneck region. The decoder module can be considered as a reconstruction module that tries to reconstruct the original data from the compressed data in the bottleneck region. Fig(1). depicts an Autoencoder architecture with an encoder on the left, bottleneck region at the center and decoder on the right.

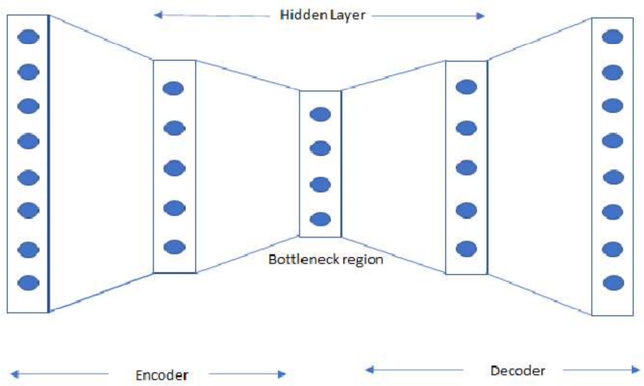

Fig (1). Auto Encoders

Equations: Let us assume an input text data X. An encoder block E converts this text to input embeddings and compresses it to lower dimension. The bottleneck layer B represents the inputs in the least possible dimension. The decoder layer D outputs embedding X'. The difference between the output and the input embeddings would be the loss in this scenario. Here we use cosine similarity to measure the loss between input and output embeddings. We use cosine similarity as the loss function since we are dealing with text data. Usually, MSE is used as the loss function for autoencoders, but in our case we have used cosine similarity to capture the semantic aspect of the text data.

$$X \sim X' \tag{1}$$

$$E = f(X) \tag{2}$$

$$B = g(E) = g(f(X)) \tag{3}$$

$$D = h(B) = h(g(f(X))) \tag{4}$$

$$Cost\ Function = \frac{1}{N}\sum_{i=1}^{N} cosinesimilarity(X, X') \tag{5}$$

From (1), we can see that X' approximates almost to X which is the primary function of autoencoder. The cost function (5) is measured as the average cosine similarity between the original input document embedding and the reconstructed document embedding. The objective is to reduce this cost function and thus increase the cosine similarity between the embeddings. During this process the autoencoder learns the important features of the text data.

Clustering: An unsupervised learning method that aims to group the input data based on the similarity of the features. Clustering has been widely used in multiple use cases where the data is unlabeled and has been found to be effective in achieving the objective of the task.

WordCloud: The documents corresponding to the embeddings grouped in the respective clusters are collected. A WordCloud is generated from the keywords of these clusters which will help identify the label of each cluster.

METHODOLOGY

In this paper, we are using an autoencoder architecture to learn the important features of the text data. The bottleneck layer of the architecture represents the text data in a lower dimension. With successful training, this layer learns the most important feature of the text data.

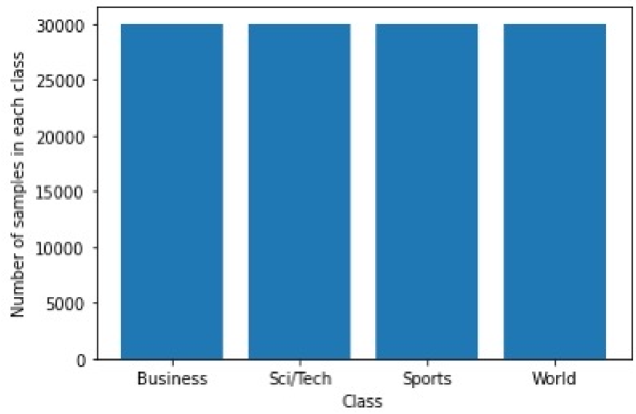

Fig. 2. Number of samples in each class

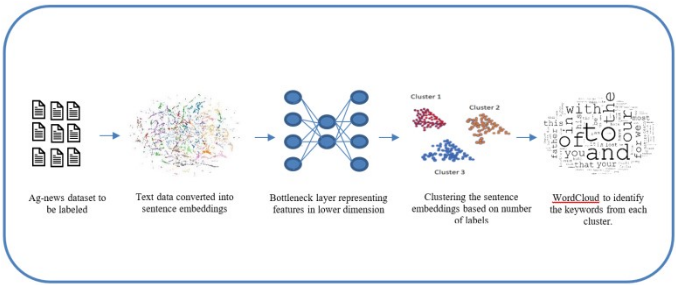

Figure 3.

Dataset: The training data considered as part of this experiment is the ag-news dataset from the Huggingface hub. The dataset is the news articles pertaining to four classes namely Business, Science/ Technology, Sports, World. It is a multiclass dataset with 30000 samples per each class as shown in Fig (2). The overall dataset consists of 120000 records.

Cleaning: As part of the pre-requisite for training process, the dataset was cleaned to remove the stop words and special words that do not add any meaning to a sentence or have an impact on the labeling activity. As part of the preprocessing step, it was inferred that the length of 75% of the training data was in the range of 30-50 tokens as show in Fig 4(a). Here the tokenization was done using normal split based on space technique. Even pre-trained BERT tokenizers can be used for the same to check if it can improve the overall performance of the system. Basic preprocessing steps were done to clean the dataset that had some special characters Fig 4(b). But certain characters like $/'= are retained Fig 4(c) to avoid losing data related to domain.

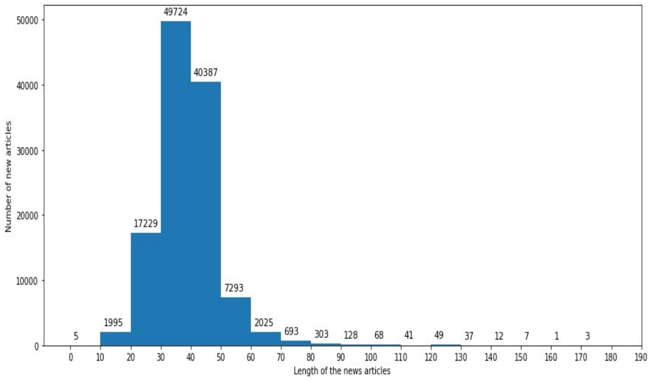

Fig. 4(a). Distribution of length of news article

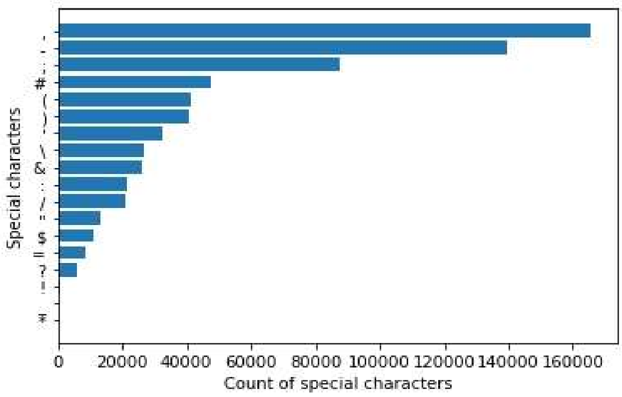

Fig. 4(b). Special characters in the dataset before cleaning

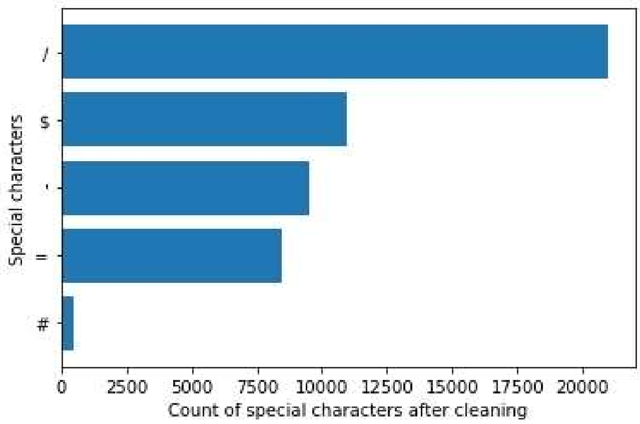

Fig. 4(c). Special characters in the dataset after cleaning

Sentence Embeddings: To represent the words/sentences in the machine understandable format we need to vectorize the input data. Here in our case our input data is a sentence and hence we must vectorize the sentence. We can achieve this vectorization through different methods like vectorizing the tokens in a document using Word2Vec [8] or Fast Text [9] and then using techniques like averaging the vectors of all the tokens in a sentence or by directly using pretrained sentence embeddings. In this modern era of deep learning, we have wide options for using sentence embeddings. Sentence Transformers is one among them that has several pre-trained models to create embeddings. For our experiments we have used pre-trained model all-mpnet-base-v2 from Sentence Transformers to create the input embeddings.

Experiment: As part of the training process, we have split the data into 80%-20% train, validation split. The training dataset consists of 96000 records and validation dataset consists of 24000 records. The training data used here is labeled. But we are proposing this solution for labeling text data when we have huge training data that is unlabeled. The training data is vectorized using the Sentence Transformer to create the sentence embeddings. The embeddings are then passed through the encoder module. In our experiment, we have used a simple undercomplete autoencoder architecture where we have bottleneck layer which represents the input in the least dimension thus retaining the important features. We have tried different combinations of hidden layers and neurons as part of our experiments as shown in Table I. A general inference from the experiment was that the results were decent when we were considering optimal dimensions in the bottleneck region rather than reducing it to very low dimensions which causes the bottleneck layer to miss capturing the most important features from the input Fig (8). We have used cosine similarity as the loss function in our experiments to measure the similarity between input and output embeddings. The cosine similarity varies between -1 to 1. Cosine similarity of -1 between two sentences indicate greater similarity between the two sentences, whereas cosine similarity of 1 indicates that two sentences are more dissimilar.

Fig. 5. Cluster 1 – World

Fig. 6. Cluster 2 – Sport

The network is then optimized to reduce this loss as part of the training process. We have used Adam optimizer with learning rate of 0.0001 for 10 epochs with batch size as 32. The weights of the bottleneck layer are learnt during the back propagation of the training phase where the model tries to learn these weights by minimizing the loss function. Once the training is completed, the entire dataset (training plus validation dataset) is passed through the encoder module and the representations are extracted from the bottleneck region. The representation from this layer is a compressed version of the input data. These representations in low dimension are then clustered using K-Means algorithm. In our case, we know that the dataset has four labels and hence we have set the number of clusters to four. While using this solution for auto labeling of text data, we can use the number of clusters same as the number of labels in case if we know about the number of labels in the dataset. In case of unknown number of labels, we can use techniques like elbow curve to identify the optimal number of labels and silhouette score to identify the measure of similarity of a data point in a cluster with other data points in the same cluster. The data points from each cluster are then fed as an input to WordCloud.

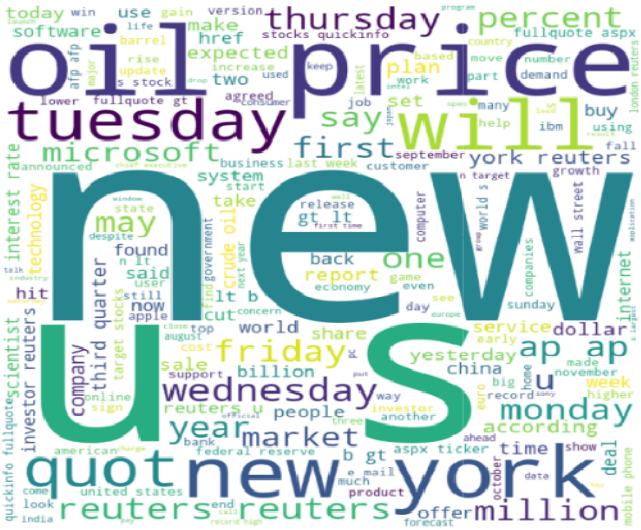

Fig. 7. Cluster 3 – Business

Fig. 8. Cluster 4 – Science/Technology

The WordCloud generates a cloud of keywords, and the size of the keywords varies depending upon the importance of those words in that cluster. Since the number of clusters in our scenario is 4, we have generated the WordCloud for these 4 clusters. As can be seen from Fig (5) showing cluster 1 contains keywords like Iraq, Iran, United States, Baghdad, Afghanistan which clearly indicates that the particular cluster is talking about world news and hence the data points pertaining to this cluster can be labeled as world in our case. Similarly, Fig (6). shows a cluster with keywords like win, season, victory, game, player which clearly indicates that this cluster is talking about sports news and hence the data points in this cluster can be labeled as sports. As can be seen in Fig (7) the keywords oil, price, billion, dollar clearly indicate that the cluster is referring to business label. From Fig (8) the keywords like Microsoft, internet, IBM, technology indicate that this cluster belongs to Science/Technology class. Table I illustrates the comparison of various iterations of the experiment. From the table, we infer that the labeling accuracy was decent when the bottle neck layer dimension was kept around 100-175. The accuracy with this setup varies from 0.34 to 0.6. When we tried to reduce the bottleneck layer dimension to very low values like below 100, it impacted the accuracy to a great level. Thus, we could infer that when the bottleneck layer dimensions decreased the autoencoders failed to capture the important features from the text data. One another parameter we tried is varying the number of hidden layers to understand how it impacted the feature extraction process. We could infer that with increase in number of hidden layers as shown in Table I, the model accuracy decreased drastically. Though the training and validation accuracies remained like other runs, but the test accuracy was impacted. From this we were able to infer that increasing the number of hidden layers increased the complexity of the autoencoder architecture thus leading to overfit. When the number of hidden layers were optimal and set to 7 and the bottle neck dimension was set to 150, the results were good. No overfitting issues were noticed during this setting run. The best accuracy of 0.6 was achieved during this run.

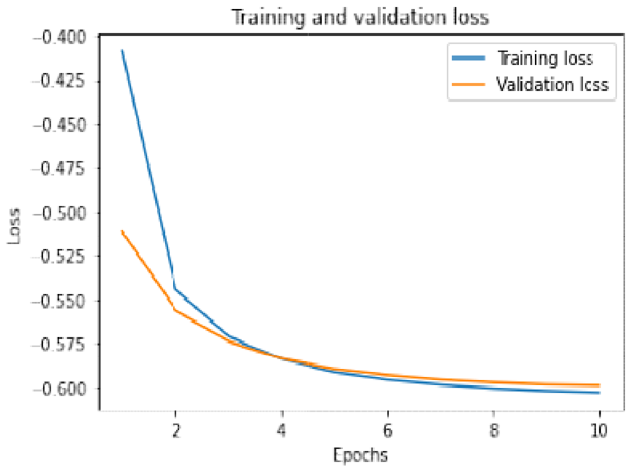

Fig. 9. Bottleneck layer dimension – 128, Number of Hidden Layers – 1

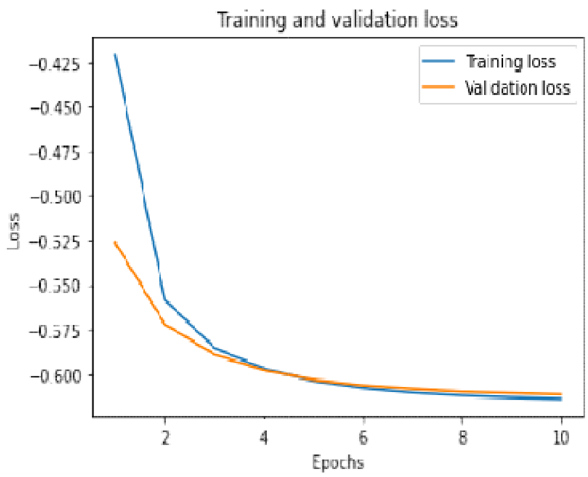

Fig. 10. Bottleneck layer dimension – 150, Number of Hidden Layers – 1

Future Work: In this paper, we have tried to address the problem of human effort required in labeling a text data even when the number of labels for the dataset is known. We have used a simple undercomplete autoencoder and clustering based technique to extract the key features

Table 1. Comparison of metrics during different iterations of the experiment

Number of Hidden Layers	Bottleneck Layer Dimension	Epochs	Class												Accuracy
			Business			Science/Technology			Sports			World			
			Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	
1	128	10	0.4	0.59	0.47	0.18	0.15	0.16	0.26	0.25	0.26	0.01	0.01	0.01	0.25
	150	10	0.37	0.55	0.45	0.95	0.69	0.8	0.29	0.24	0.26	0.33	0.31	0.32	0.45
	175	10	0.23	0.2	0.21	0.95	0.67	0.79	0.35	0.52	0.42	0.34	0.31	0.33	0.43
6	16	10	0.03	0.03	0.03	0	0	0	0.46	0.74	0.57	0.03	0.02	0.03	0.2
7	150	10	0.88	0.75	0.81	0.88	0.9	0.89	0.32	0.34	0.33	0.37	0.39	0.38	0.6
9	100	10	0.01	0.01	0.01	0.94	0.68	0.79	0.39	0.61	0.47	0.07	0.05	0.06	0.34
9	150	10	0.05	0.05	0.05	0.01	0.01	0.01	0.37	0.45	0.41	0.02	0.01	0.01	0.13
21	10	10	0.09	0.25	0.14	0.21	0.06	0.09	0	0	0	0.2	0.2	0.2	0.13

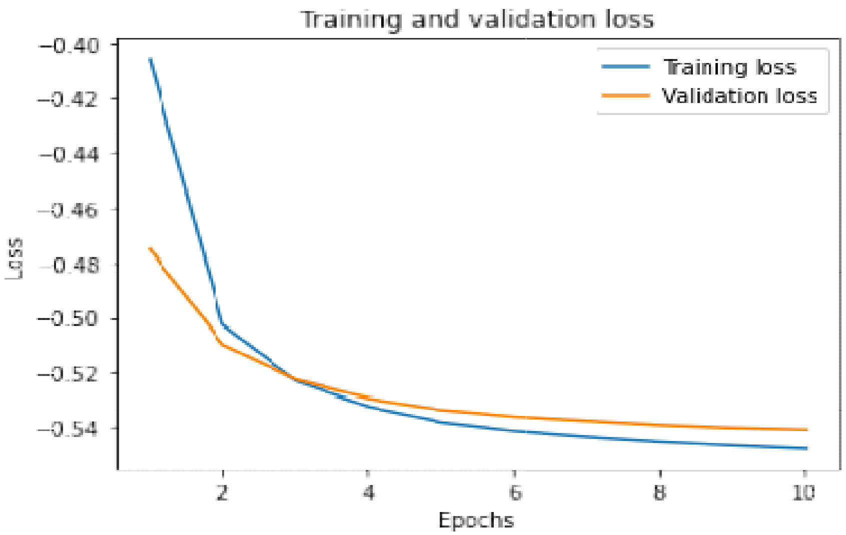

Fig. 11. Bottleneck layer dimension – 150, Number of Hidden Layers – 7

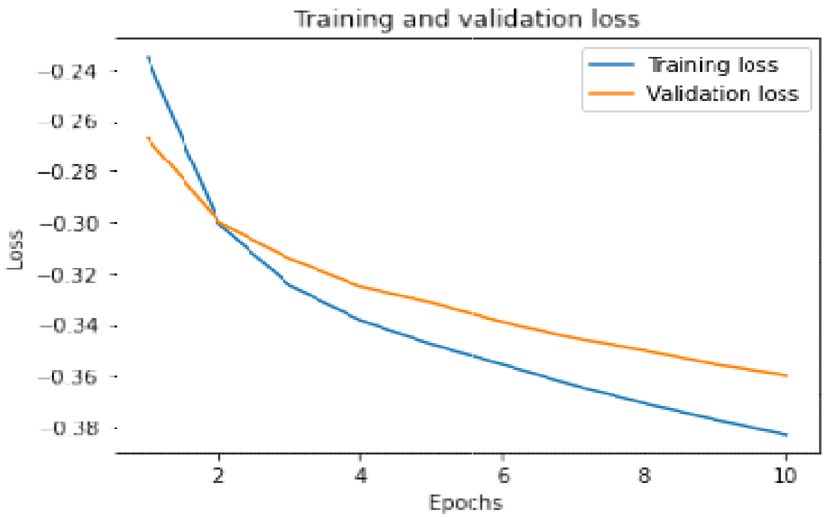

Fig. 12. Bottleneck layer dimension – 10, Number of Hidden Layers – 21

and cluster them to create labels. We have run experiments with different iterations by varying the neuron size and hidden layer size and keeping the number of epochs constant. Our initial objective was to understand how efficient we can use this technique for labeling an unlabeled text dataset and we have achieved 0.6 accuracy using it. The primary scope of future work is to try this architecture for a domain specific dataset (eg. Banking, Insurance) to understand how well domain knowledge can be captured using this technique, as there will be limitations like limited labeled data for domain specific data. We would also like to extend the scope of this work to try different embedding techniques to generate the input for the autoencoder.

REFERENCES

[1] Paraskevi Nousi, Anastasios Tefas," Self-supervised autoencoders for clustering and classifcation", Springer-Verlag GmbH Germany, part of Springer Nature 2018

[2] Soodeh Hosseini, Zahra Asghari Varzaneh," Deep text clustering using stacked AutoEncoder", Springer Science+Business Media, LLC, part of Springer Nature 2022

[3] Dor Bank, Noam Koenigstein, Raja Giryes," Autoencoders," arXiv:2003.05991v2 [cs.LG] 3 Apr 2021

[4] Munmi Gogoi, Shahin Ara Begum, "Image Classification using Deep Autoencoders", 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)

[5] Zhenyu Yang, Xue Pang, " Research on Text Classification of Denoising Autoencoder Based on Additional Momentum and Adaptive Learning Rate", 2018 11th International Symposium on Computational Intelligence and Design (ISCID)

[6] Chaman Lal et al., "Text Clustering using K-MEAN " , International Journal of Advanced Trends in Computer Science and Engineering, 10(4), July – August 2021, 2892 – 2897 2892

[7] Nils Reimers, Iryna Gurevych," Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", EMNLP 2019

[8] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, "Efficient Estimation of Word Representations in Vector Space", arXiv:1301.3781v3 [cs.CL] 7 Sep 2013

[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov, "Enriching Word Vectors with Subword Information", arXiv:1607.04606v2 [cs.CL] 19 Jun 2017

****