Handling Imbalanced Data using UP-Sampling and Data Augmentation for NLP

N Vidya Sagar
Computer Science and Engineering
Chalapathi Institute of Engineering and Technology
Lam, Guntur-34

P Venkata Siva
Computer Science and Engineering
Chalapathi Institute of Engineering and Technology
Lam, Guntur-34

***Abstract*— This paper explores different options to deal with data insufficiency (smaller dataset) and class imbalance problems when dealing with NLP using Deep learning techniques. In developing applications like auto assigning of service tickets to departments using NLP and Deep Learning techniques some of the commonly faced challenges are a smaller number of observations(tickets) and some departments having a smaller number of tickets which leads to class imbalance. This makes it impossible to make use of cutting-edge deep learning techniques. This is an attempt to compare different approaches to up sample and augment the data to build deep-learning based models (e.g., LSTM) and compare the results against the original smaller data set. This paper also provides general guidance into what data augmentation techniques to use to improve model performance with smaller and imbalanced datasets.**

I. INTRODUCTION

Applications of deep learning in NLP has achieved greater heights in the past few years. Introduction of techniques like LSTM[1] and transformers has changed the landscape of solving NLP problems. The biggest challenge these techniques have is they depend on large training datasets suffer from data im-balance. In many cases having large training datasets is not an option due to small scale operations. In Assigning tickets received thru emails and online applications to respective departments in an industry is a daunting and tedious task. Many NLP and Data science models are built to semi automate the process by building classification models to auto tag the tickets. While training these type of classification models the common challenges are smaller datasets and few departments have far less data and leads to under trained models and poor performance. For structured data there are a wide variety of techniques like SMOTE to up sample or down sample to the data to create datasets that are of good size and enough class balance. For text data the problem is many folds, a simple up sampling the data might not work. To overcome these problems a few data augmentation techniques are introduced. This is an attempt to use

different kinds of augmentation techniques along with data up sampling on a smaller dataset to build deep learning-based classification models and benchmark them against the original smaller im-balanced dataset designations.

II. BACKGROUND

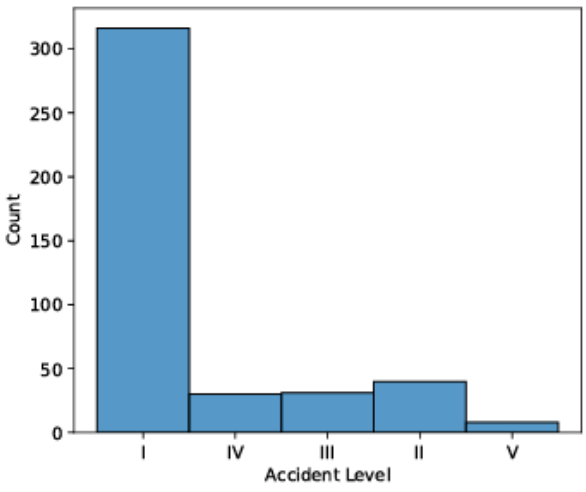
A lot of text classification problems in the industry have problems with less volume of data and class imbalance. For example, customer emails and tickets data is too difficult to classify due to less volume class imbalance. Need right data up-sampling or augmentation techniques to handle such problems. Emails and tickets are considered very confidential (PII) and not publicly available to try any deep learning models. For this purpose, a dataset resembling the customer tickets data with both the problems, i.e., smaller in size and having class imbalance is chosen to build a classifier and compare the results with augmentation.

The dataset[2] comes from one of the biggest industries in Brazil and in the world. It is an urgent need for industries/companies around the globe to understand why employees still suffer some injuries/accidents in plants. Sometimes they also die in such an environment. There are mainly two columns in the dataset

Description: “Detailed description of how the accident happened”

Accident level: “The severity of the accident (I – V)”

along with few other columns like gender, country etc. The dataset has only 425 records with class -imbalance which makes it very challenging to build NLP based classifiers using deep learning.



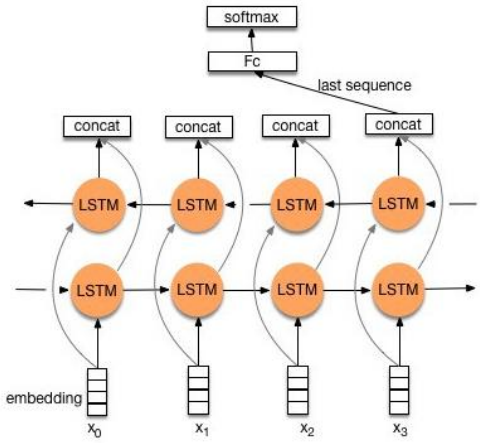
III. APPROACH

The Approach discussed in the paper is to first divide the data into standard train and test splits. Develop a classifier using Bi-LSTM[3] to measure the model performance, this will act as the benchmark to compare against the methods. In the next attempt up-sample and augment the data (only the train portion) using data augmentation techniques, and build a similar (Bi-LSTM) based model to compare the performance with the original dataset. Also try a simple up sampling technique to increase the volume of the data (train portion

only) and build a classifier to compare against the benchmark and finally conclude the best approach using the performance metrics.

A. *Why Bi-LTSM?*

Bi-directional layer can do a reverse flow of information as well. In problems where all timesteps of the input sequence are available, Bidirectional LSTM train two instead of one LSTMs on the input sequence. This can provide additional context to the network the first on the input sequence as is and the second on a reversed copy of the file and result in faster and even fuller learning on the problem.



B. Model *evaluation*

Choosing an appropriate metric is challenging and also it is difficult for imbalanced classification problems.

There are standard metrics that are widely used for evaluating classification models, such as classification accuracy or classification error.

Accuracy = Correct Predictions / Total Predictions

Error = Incorrect Predictions / Total Predictions

This challenge made it even more difficult when there is a skew in the class distribution(im-balance). The reason for this is that many of the standard metrics become unreliable when classes are imbalanced.

Imbalanced classification problems typically rate classification errors with the minority class as more important than those with the majority classes. As such performance metrics may be needed the focus on the minority classes, which is made challenging because it is the minority classes where we lack observations required to train an effective model.

The F1-Score is a popular metric for imbalanced classification.

F-Measure = (2 * Precision * Recall) / (Precision + Recall)

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

IV. DATA CLEANING AND PRE-PROCESSING

Following pre-processing steps were done on the description column:

Converted text to lowercase: this is to ensure that all the words are in the same case, so that we don't miss out on any words due to case mismatch.

Removed special characters: remove punctuation marks, brackets etc which don't contribute much.

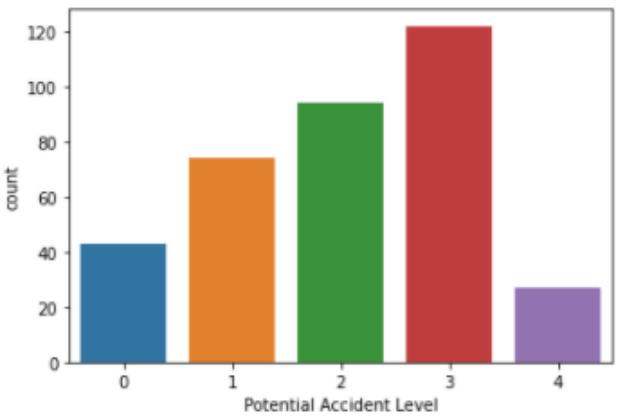
Removed numerical data: removed all digits from 0-9 occurring in the Description.

Removed stopwords using nltk: This will remove frequently occurring words like the, in, a, an etc. Applied Lemmatization, using the WordNetLemmatizer, on the words to get the base word.

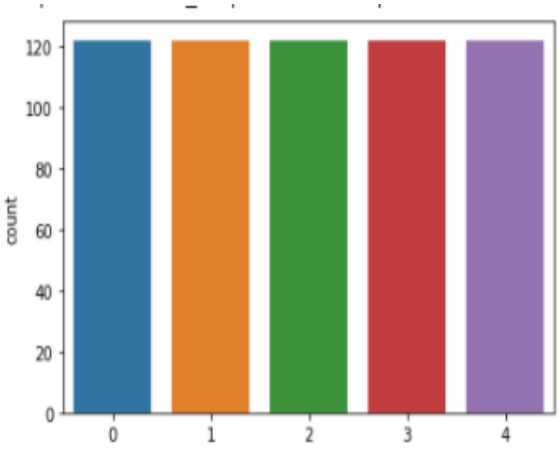
A. *Data Up-Sampling*

SMOTE technique (synthetic minority oversampling technique) is one of the most commonly used oversampling methods to solve the imbalance problem. It aims to balance class distribution by randomly increasing minority class examples by replicating them. SMOTE is applied on the training data.

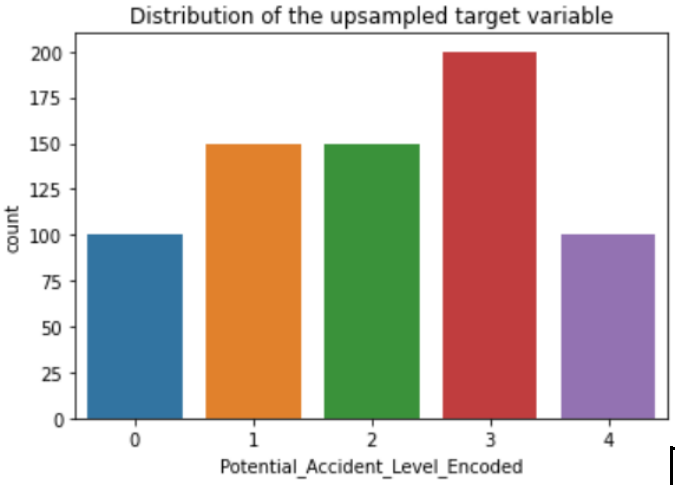
Before Up-sampling There are 424 entries in the dataset which is split into 360 record for training and 64 records into testing



1) SMOTE: After Up-sampling Using SMOTE [4] train data is increased to 610 records



2) Random up-sampling: Training set has 700 records



3) Data Augmentation using Random Shuffling: The method of data augmentation used is random shuffling using the RANDOM[5] library import.

The original data frame of shape (424,2) is split into training and test by taking the first 360 rows as training data and the next 64 rows as testing data.

The training data is then passed into the below function to get the shuffled Description column.

The shuffled data is then appended to the original training data dataframe. The new dataframe shape is now (720,2).

V. MODELLING AND RESULTS

A. Bi-Directional LSTM with Glove Word Embeddings:

The Glove[6] Word Embeddings used above for LSTM is used for Bi-Directional LSTM as well.

The above was then provided as input to the Embedding Layer of the Network.

```
model2 = Sequential()
model2.add(Embedding(vocab_size,200, weights = [embedding_matrix]))
model2.add(Bidirectional(LSTM(200,return_sequences = True)))
model2.add(GlobalMaxPool1D())
model2.add(Dropout(0.2))
model2.add(Dense(200,activation = 'relu'))
model2.add(Dropout(0.2))
model2.add(Dense(5,activation = 'softmax'))
```

The model is then trained on the Original, SMOTE, Manually Up sampled data and Augmented data using epochs = 10 and batch size = 32.

Model on the Cases (Glove)	Accuracy
Bi-Directional LSTM on Original Data	56.25%
Bi-Directional LSTM on SMOTE Data	45.3125%
Bi-Directional LSTM on Manually Up sampled Data	93.75%
Bi-Directional LSTM on Augmented Data	96.875%

B. *Bi*-Directional *LSTM with Word2Vec Word Embeddings:*

The Word2Vec[7] Word Embeddings used above for LSTM is used for Bi-Directional LSTM[8] as well.

The above was then provided as input to the Embedding Layer of the Network.

```
model3 = Sequential()
model3.add(Embedding(vocab_size,300, weights = [embedding_matrix_wv]))
model3.add(Bidirectional(LSTM(200,return_sequences = True)))
model3.add(GlobalMaxPool1D())
model3.add(Dropout(0.2))
model3.add(Dense(200,activation = 'relu'))
model3.add(Dropout(0.2))
model3.add(Dense(5,activation = 'softmax'))
```

The model is then trained on the Original, SMOTE, Manually Up-sampled data and Augmented data using epochs = 10 and batch_size = 32.

The accuracies are as below

Model on the Cases (Word2Vec)	Accuracy
Bi-Directional LSTM on Original Data	45.3125%
Bi-Directional LSTM on SMOTE Data	46.875%
Bi-Directional LSTM on Manually Up-sampled Data	93.75%
Bi-Directional LSTM on Augmented Data	95.3125%

VI. LIMITATIONS

Tried on a small limited dataset, need to repeat similar experiments with different datasets to arrive at effectiveness of each technique.

VII. CONCLUSION

From the above results it is observed that both manually up-sampled set and Augmented dataset gave better results compared the original dataset and up-sampled dataset with SMOTE. A combined approach can be derived for improving the results and make a standard process to handle less data and class imbalance in text mining problems.

REFERENCES

[1] Has¸im Sak, Andrew Senior, Francoise Beaufays, Google, USA, Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling

[2] dataset : https://www.kaggle.com/ihmstefanini/industrial-safety-and-health-analytics-database

[3] Zhiheng Huang, Wei Xu, Kai Yu, Bidirectional LSTM-CRF Models for Sequence Tagging, Aug 2015

[4] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer , SMOTE: Synthetic Minority Over-sampling Technique , 2002

[5] https://amitness.com/2020/05/data-augmentation-for-nlp/

[6] Jeffrey Pennington, Richard Socher, Christopher D. Manning, GloVe: Global Vectors for Word Representation

[7] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Efficient Estimation of Word Representations in Vector Space, 2013