Introduction to NLP
Final Report

Team Name : Edge
Team No. : 43
Project Title : Measure Text Fluency (3)
Team Members : Sankalp Thakur (2021201042)
Shravan Sharma (2021201058)
Vineet Agrawal (2021201049)

1. Introduction:

The focus of our project "Measuring Text Fluency." The significance and relevance of resolving this issue have significantly expanded as a result of the world's ever-expanding literary content.

The term "text fluency" relates to an estimation of a text's quality, readability, accuracy, and comprehension. It goes without saying that any text produced by an automatic (non-human) source is likely to have linguistic faults and needs to be manually corrected. To address this issue, measuring text fluency is crucial since it shows whether the necessary information is presented in a desirable way from a given piece of text. This necessitates considering a variety of factors, including grammar, style, word choice, etc.; as a result, it is a demanding and difficult undertaking.

Our approach involves a multi-class classification problem, classifying a piece of text as either Not Fluent, Neutral, or Fluent. Our work has involved conducting a literature survey to understand previous approaches, implementing a baseline, conducting experiments using supervised machine learning algorithms, and analyzing the results in detail.

2. Dataset:

We worked on the dataset [Microsoft Compression Dataset](). Each data sample is either sentences or short paragraphs with up to five corresponding shorter (compressed) versions. Along with this, this dataset contains average meaning and arerage grammar score for each data sample. This score $\epsilon\ [1, 3]$.

We have considered the fluency of text as weighted sum of both meaning and grammar score and then rounded off this value to get closest integer value.

To get weight values we used different evaluation metrics and evaluated average correlation between metrics and fluency labels with different weights.

```
averageMeaning weight : 0.0 , averageGrammar weight : 1.0  correlation to fluency : 0.08835025225926323
averageMeaning weight : 0.1 , averageGrammar weight : 0.9  correlation to fluency : 0.08983207992680046
averageMeaning weight : 0.2 , averageGrammar weight : 0.8  correlation to fluency : 0.09025994831472317
averageMeaning weight : 0.3 , averageGrammar weight : 0.7  correlation to fluency : 0.08918458219569154
averageMeaning weight : 0.4 , averageGrammar weight : 0.6  correlation to fluency : 0.08625491168806287
averageMeaning weight : 0.5 , averageGrammar weight : 0.5  correlation to fluency : 0.08138502484553721
averageMeaning weight : 0.6 , averageGrammar weight : 0.4  correlation to fluency : 0.07484737103806909
averageMeaning weight : 0.7 , averageGrammar weight : 0.3  correlation to fluency : 0.06720266385791064
averageMeaning weight : 0.8 , averageGrammar weight : 0.2  correlation to fluency : 0.05910413484254443
averageMeaning weight : 0.9 , averageGrammar weight : 0.1  correlation to fluency : 0.05110903357203279
averageMeaning weight : 1.0 , averageGrammar weight : 0.0  correlation to fluency : 0.04358919350254622
```

We found that 0.2 for average meaning score and 0.8 for average grammar score had highest positive correlation even though correlation is weak.

$$Fluency\ =\ closest\_integer(0.2\ *\ meaning_{score}\ +\ 0.8\ *\ grammar_{score})$$

We rounded the fluency to its nearest whole value. This categorized our dataset into 3 classes.This way we converted our problem into a classification problem.

Fluency Score	Represents	No. of Sample
0	Not-fluent	127
1	Neutral	3352
2	Fluent	19402

3. Methodology:

We did minimal preprocessing on text.We converted whole data lower-case and avoided tampering with data as it might lead to changing the fluency of the sentence. Punctuations are an important part of text fluency. Similarly, stop words are also important. Removing or changing the form of words will also change the fluency of the sentence. We tokenized the data splitting the sentences into words. We then converted the sentences into sequences of numbers to uniquely identify each token. Two token ids were reserved for padding and unknowns. Any OOV token is given token id 1 while the data was pre padded with zeros to make every sentence of equal size.

The data that was available was highly unbalanced. To balance the data we used 3 different samplers and trained models on the original data as well balanced datasets.

Class Balancing:

1. SMOTE : This method replicates minority class examples to increase their number at random in an effort to balance the distribution of classes. SMOTE combines already existing minority instances to create new minority instances. For the minority class, it creates fresh training samples by linear interpolation. In other words, these artificial training records are created by randomly choosing one or more of each example's k-nearest neighbours, performing the appropriate linear operations, and raising the minority sample size.

2. BorderlineSMOTE : This algorithm is a variant of the original SMOTE algorithm. Borderline samples will be detected and used to generate new synthetic samples.

3. ADASYN : This method is similar to SMOTE but it generates different number of samples depending on an estimate of the local distribution of the class to be oversampled

We used 4 methods for classification.

- Non-DL classifiers
- Non-DL classifiers using POS Tagging
- LSTM using GLove Embeddings
- BERT Encoding

For every method we first trained the model on unbalanced data and then tried the oversamplers to train the classifiers on balanced data.

1. **Non-DL classifiers:**We first used traditional classifiers on the data. 4 classifiers were used:
    1. Logistic regression
    2. SVM
    3. Random Forest
    4. Xgboost
2. **Non-DL classifiers with POS Tagging:**We converted the sentences into Part of speech tokens and then converted these tokens into sequence of numbers.
   Again we used the same classifiers for training as used in the previous method.
3. **LSTM with GLove embeddings:** We created a model with embeddings in the first layer and the two layers of LSTM followed by multiple fully connected layers. The last layer

contains 3 units with softmax activation. The embedding layer wawa loaded with pre-trained Glove embeddings for available words and random vectors for other words.

4. **FCNN with Bert embeddings:**We used bert get vector encoding of the words of sentence and then calculated the mean of the vectors representing the sentence encoding. This fixed size vector was then fed to a multilayer Fully connected Neural network. The last layer contains 3 units with softmax activation

4. Analysis:

Non-DL classifiers

Logistic regression

	Precision	Recall	Accuracy	F1-score
None	**0.48**	**0.35**	**0.85**	**0.34**
Smote	**0.35**	**0.37**	**0.57**	**0.32**
Borderline	**0.35**	**0.36**	**0.62**	**0.33**
Adasyn	**0.36**	**0.38**	**0.58**	**0.32**

SVM

	Precision	Recall	Accuracy	F1-score
None	**0.48**	**0.35**	**0.85**	**0.35**
Smote	**0.44**	**0.34**	**0.84**	**0.34**
Borderline	**0.37**	**0.36**	**0.81**	**0.34**
Adasyn	**0.40**	**0.35**	**0.83**	**0.34**

Random Forest

	Precision	Recall	Accuracy	F1-score
None	**0.49**	**0.37**	**0.86**	**0.37**
Smote	**0.41**	**0.38**	**0.83**	**0.39**
Borderline	**0.41**	**0.38**	**0.83**	**0.39**
Adasyn	**0.41**	**0.38**	**0.83**	**0.39**

XGBOOST

	Precision	Recall	Accuracy	F1-score
None	**0.47**	**0.37**	**0.85**	**0.38**
Smote	**0.38**	**0.38**	**0.79**	**0.38**
Borderline	**0.39**	**0.39**	**0.80**	**0.39**
Adasyn	**0.38**	**0.38**	**0.80**	**0.38**

Logistic regression and SVM perform poorly on data. Tree based algo performed better. While the accuracy is comparable, both XGBoost and Random Forest have better metrics than logistic regression and svm.Random Forest have better average precision while XGBOOST has better recall when oversampled using borderline.

POS Tagging

Logistic regression

	Precision	Recall	Accuracy	F1-score
None	**0.28**	**0.33**	**0.85**	**0.31**
Smote	**0.34**	**0.33**	**0.42**	**0.26**
Borderline	**0.34**	**0.35**	**0.53**	**0.30**
Adasyn	**0.33**	**0.32**	**0.42**	**0.26**

SVM

	Precision	Recall	Accuracy	F1-score
None	**0.28**	**0.33**	**0.85**	**0.31**
Smote	**0.33**	**0.34**	**0.69**	**0.33**
Borderline	**0.34**	**0.35**	**0.58**	**0.32**
Adasyn	**0.34**	**0.35**	**0.55**	**0.31**

Random Forest

	Precision	Recall	Accuracy	F1-score
None	**0.49**	**0.37**	**0.86**	**0.38**
Smote	**0.46**	**0.39**	**0.86**	**0.40**
Borderline	**0.46**	**0.38**	**0.85**	**0.40**
Adasyn	**0.46**	**0.38**	**0.85**	**0.40**

XGBOOST

	Precision	Recall	Accuracy	F1-score
None	**0.48**	**0.39**	**0.86**	**0.39**
Smote	**0.42**	**0.39**	**0.83**	**0.40**
Borderline	**0.42**	**0.39**	**0.84**	**0.40**
Adasyn	**0.42**	**0.39**	**0.83**	**0.40**

Again we see a similar trend where Tree based algorithms outperform other 2 algorithms. Using POS we get slightly better results when compared to the previous method. We achieve the highest F1 score of 0.40 in multiple sampling techniques.

Glove

	Precision	Recall	Accuracy	F1-score
None	**0.53**	**0.37**	**0.86**	**0.38**
Smote	**0.46**	**0.38**	**0.85**	**0.39**
Borderline	**0.46**	**0.38**	**0.85**	**0.39**
Adasyn	**0.48**	**0.39**	**0.85**	**0.40**

BERT

	Precision	Recall	Accuracy	F1-score
None	**0.34**	**0.33**	**0.85**	**0.31**
Smote	**0.35**	**0.38**	**0.45**	**0.29**
Borderline	**0.36**	**0.38**	**0.61**	**0.35**
Adasyn	**0.35**	**0.38**	**0.46**	**0.29**

LSTM with GLove performed better, just like XGBoost and Random Forest. Best model was trained when using Adasyn sampling with following metrics.
Precision - 0.48 Recall - 0.39 and F1 score - 0.40 is best combination for model.Bert Performed Poorly. Possible Reason could be instead of taking individual word embeddings, mean of the word embeddings was assumed to be sentence embeddings.

5. Code:

The code can be found here

6. Conclusion:

The dataset was highly imbalance and could not be efficiently divided into classes. A better dataset required which has higher number of non fluent samples which are also distinguishable is required to achieve the task of fluency classification.Over sampling helped to an extent but was not very beneficial. LSTM worked best for the current data. Tree based algorithms also performed good.