Team 27: Measure Text Fluency

Thota Gokul Vamsi, 2019111009

Sai Akarsh C, 2019111017

Abstract

This report consists of the detailed description of the entire work done as a part of this project - relevant literature, ideas / experiments, results and error analysis.

1. Introduction

The project we are working on is 'Measuring Text Fluency'. The importance and relevance of addressing this problem has increased to a significant extent, due to ever-growing amount of textual content in the world.

Text fluency refers to an estimate which corresponds to the quality, readability, accuracy and comprehensibility of a piece of text. It is an obvious fact that any such textual information generated by an automated (non-human) source is prone to errors of language understanding, and requires manual rectification. Measuring text fluency is an important task to address this issue, as it describes if the required information is conveyed in a desirable format, from a given piece of text. This requires taking into account various criteria such as grammar, style, choice of words etc. and hence is a relatively challenging and non-trivial task.

We address this problem as a multi-class classification problem, where a piece of text is classified either as Not Fluent, Neutral or Fluent. We also use some methods which take manual references and annotations into account, for measuring fluency. We have worked on understanding different approaches which were used to address this challenge, by conducting a thorough literature survey, implemented the baseline and performed various experiments using well-known supervised machine learning algorithms. Further, we have explored neural networks (RNNs, LSTMs) and transformers, for language modelling. Eventually, we performed a detailed analysis of the observations, and developed a simple application which measures text fluency in real time.

2. Dataset

We worked with the Microsoft Compression Dataset which contains about 23k data samples, where each sample contains a piece of text (1-2 sentences or a paragraph) and a manually added reference, to verify the fluency score of the corresponding text. Each sample was further manually annotated on a scale of 1 to 3, where 3 meant very fluent and 1 meant not fluent - and this fluency score was computed for both meaning and grammar of the sentence.

We have considered a weighted average of the both annotations, by assigning a weightage of 0.7 to grammar and 0.3 to meaning of the text, thus achieving a single score for every data sample. This score was further rounded off to nearest integer, which acts as the label of the sample. If this value was 1, it was considered to be 'non-fluent', a score 2 indicated 'neutral' and 3 indicated that it was 'fluent'. This has hence been approached as a 3-class classification problem, for validating various ideas and experiments.

3. Literature review

Liu et al. [4] have explored an approach to measure text fluency of an MT system, and have taken into account the n-gram features for scoring a sentence based on its fluency. It was also found that there was a high correlation between such features and fluency, thus achieving decent results even in absence of manual references for given text. Kann et al. [2] have proposed a measure called syntactic log-odds ratio (SLOR), which acts as a normalized LM scoring to estimate fluency. They have also incorporated wordpiece models to reduce vocabulary size and improve unknown-words handling.

Dusek et al. [1] have tried a supervised-learning approach using LSTMs to predict the quality of text. Lin et al. [3] have attempted using metrics such as ROUGE-L (longest common subsequence length between a sentence and its manual reference), and ROUGE-S (Skip-Bigram matches between a sentence and its manual reference) for evaluating quality of an MT system. BLEU score [5] was

also considered to be a reliable metric for understanding quality of text and evaluating translated text's quality, by Yonghui et al. [6].

4. Methodologies

For the baseline, we have implemented an n-gram language model (n = 3) which is trained on the train set, and is used to obtain probability scores for all n-grams in a sentence, and uses few highest and least probability values as features as done in this paper. Logistic Regression is used to perform the 3-class classification based on these features. Further, different scoring functions used in the work done by Liu et al. are also utilized to compute the correlations of necessary features.

Regarding further experiments, textual features, POS-features, different Language models (LSTM, RNN, BERT), hyper-parameter tuning, along with special analysis of a selected list of features and their correlation with fluency, were also conducted. The ideas, experiments and techniques which were implemented are enumerated below:

- Attempted different values for parameters such as number of features considered, n in n-grams, probability thresholds used in scoring functions, i.e, considering only those probability features for a piece of text which are greater than / less than a certain threshold - for fluency prediction.
- Extracted textual features alone (using Tf-idf vectorizer) to understand how these features play a role in classification. Previously mentioned classifiers were employed here too.
- Implemented addition of POS (part of speech) n-gram features, and performed classification based on it using machine learning approaches.
- Different language models in place of the ngram language model, such as using Recurrent Neural Networks and Long Short-Term Memory networks.
- Transformer language models such as BERT for calculating the perplexity and probability related features for a given sentence.
- A simple application which takes in a piece of text as input and predicts its corresponding label as output (Fluent / Not Fluent / Neutral) in real time.

After obtaining data from the required dataset, we analyzed it and applied appropriate pre-processing as necessary. One challenge with the dataset is that it is highly skewed with respect to the labels, where very few samples correspond to non-fluent texts and majority of samples correspond to fluent texts. For tackling class imbalance, we have tried two different over-sampling approaches which balance the class distribution - Random over sampling and SMOTE (Synthetic Minority Over-sampling Technique). They are described below.

4.1. Data sampling

- **Random Oversampling**: Random oversampling involves randomly duplicating examples from the minority class and adding them to the training dataset. Examples from the training dataset are selected randomly with replacement. This process can be repeated until the desired class distribution is achieved (in this case, there should eventually be equal number of examples corresponding to each class).
- **Synthetic Minority Over-sampling Technique (SMOTE)**: This technique aims to balance class distribution by randomly increasing minority class examples by replicating them. SMOTE synthesizes new minority instances between existing minority instances. It generates new training samples by linear interpolation for the minority class. In other words, these synthetic training records are generated by randomly selecting one or more of the k-nearest neighbors for each example in the minority class, and applying linear operations in the desired way, thereby increasing minority samples.

Data was initially shuffled, and split in a 70-15-15 ratio for training, validation and testing respectively. Different metrics like accuracy, precision, recall, f1-score, confusion matrix etc. were logged for analysis of a model's / scoring function's performance under different scenarios with respect to data sampling.

4.2. Metrics for fluency

The different metrics (features) which were examined closely regarding their relevance in predicting fluency, are described in detail below:

- **BLEU Score**: BLEU score (Bilingual evaluation understudy) is primarily utilized to evaluate the quality of machine translated output, and comparing it with human annotations. It is known to highly correlate with human judgement of quality. It considers the N-grams involved in the sentence output and compares it with the human reference annotation, and computes a score based on a complex mathematical formulation. A score between 0 and 1 is eventually obtained, where 0 indicates no similarity and 1 indicates very high similarity. This idea was explored as it can be useful in understanding if the provided text does significantly

overlap with reference provided by expert annotators, and that could be positively correlated with fluency.

- **ROUGE-S**: This metric corresponds to skip-ngram co-occurrence statistics. All the subsequences of a fixed length (n in ngram) are considered in both the provided text and its corresponding human translation, and their overlap is measured. This is further mathematically manipulated to obtain a score between 0 and 1, where the score is directly proportional to the overlap / similarity between the text and the reference. Thus, high score indicates higher similarity value, which in turn indicates more fluency as it aligns well with human judgement.
- **ROUGE-L**: This metric has a similar intent as that of the above metric, where the idea of a subsequence is used. For both the provided text and the reference annotation, the length of the longest common subsequence between both of them is considered as the value for this metric. The primary intent is that the larger the value, the better the alignment with human annotation, and hence, higher the fluency.
- **SLOR**: SLOR (Syntactic Log-odds ratio) is independent of the human translation, and primarily depends only on the language model's assigned probability for the sentence. It is normalized such that uncommon words which occur in a similar context (such as France and Tuvalu - both being places occur in similar context, but one more frequent than the other) are not penalized by the language model. As it is a function of the probability of the occurrence of the sentence, higher probability indicates adherence to grammatical rules and thus fluency.
- **N-gram overlap**: This metric is similar to ROUGE-S except for the fact that instead of fixed length subsequences, fixed length contiguous sub-segments are considered in both the provided text and its reference annotation, and an intersection over union score is computed. This score clearly lies between 0 and 1, and is proportionate to alignment of reference with the text, and thus the fluency.

5. Analysis

The stats and their corresponding analysis, for each idea / experiment implemented, are explained below.

5.1. Baseline

As previously explained, the baseline approach used for this paper, primarily relies on a trained n-gram language model and n-gram probabilities of a sentence, where n is a pre-determined value. To fix the number of features, it was implemented such that only K highest and K least values of probabilities would be used as features, where K is chosen based on analysis of the data and lengths for each piece of text / sample in the data. The values of n and K are chosen to be 3 and 12 respectively, based on observations from the data. These probability features along with perplexity scores are used in the classification process. The results achieved on using different sampling techniques (without any oversampling, Random Oversampling, SMOTE) can be viewed below. This classification is highly affected by the bias in the dataset, leading to majority class / single class prediction (which explains low macro accuracy metrics).

Type	Accuracy	Recall	Precision	F1-score
None	0.871	0.333	0.290	0.310
Random Over	0.871	0.333	0.290	0.310
SMOTE	0.126	0.333	0.041	0.075

Further, there are 2 different scoring functions which were implemented to observe their correlation with fluency. The first scoring function simply considers the sum of log probabilities for all n-grams in the text, divided by number of tokens in the text (to normalize by text length). The Pearson correlation score of this function was found to be 0.0114 approximately, which indicates a positive correlation value between probabilities of an n-grams' occurrence and fluency.

The second scoring function considers only those probability values, if they either represent a rarely occurring n-gram (probability below a threshold) or a relatively frequently occurring n-gram (probability higher than a different threshold). The Pearson correlation score of this function was found to be -0.002 which indicates a negative weaker correlation, due to fact that the score is actually scaled by probability values based on the thresholds they satisfy.

5.2. Different parameters in N-gram approach

Sampling Type	n	K	Accuracy	Recall	Precision	F1-score
None	3	12	0.871	0.29	0.333	0.31
None	3	16	0.871	0.29	0.333	0.31
None	4	12	0.871	0.29	0.333	0.31
None	4	16	0.871	0.29	0.333	0.31
Random Over	3	12	0.871	0.29	0.333	0.31
Random Over	3	16	0.871	0.29	0.333	0.31
Random Over	4	12	0.871	0.29	0.333	0.31
Random Over	4	16	0.871	0.29	0.333	0.31
SMOTE	3	12	0.126	0.042	0.333	0.075
SMOTE	3	16	0.126	0.042	0.333	0.075
SMOTE	4	12	0.126	0.042	0.333	0.075
SMOTE	4	16	0.126	0.042	0.333	0.075

Here n is the N-gram size used, K is used in calculating the feature vector. The feature vector is made of k most and least frequently occurring N-gram probabilities.

From the results obtained above, it can be observed that there is not much change on varying n and K parameters. This indicates that the model used for feature extraction is not able to capture the details properly. We can see a significant reduction in performance when we use SMOTE as the sampler.

5.3. Supervised methods on textual features

Different supervised machine learning algorithms such as Decision Trees, Random Forest, Support Vector Machine, K-nearest neighbours were implemented by taking simply the textual features related to word counts and scores into account. Tf-idf vectorizer was utilized to obtain vectors for each data sample, where each dimension corresponds to a word in the vocabulary. The vocabulary size was fixed to be 6000 due to computational constraints. The results achieved on using different sampling techniques (without any oversampling, Random Oversampling, SMOTE) for each algorithm can be viewed below.

K-nearest neighbours

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.875	0.333	0.292	0.311
Test (None)	0.872	0.333	0.291	0.31
Train (Random Over)	0.645	0.645	0.651	0.592
Test (Random Over)	0.233	0.325	0.335	0.167
Train (SMOTE)	0.664	0.664	0.494	0.552
Test (SMOTE)	0.124	0.344	0.046	0.081

Decision Tree

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.893	0.564	0.667	0.604
Test (None)	0.789	0.361	0.358	0.359
Train (Random Over)	0.913	0.913	0.927	0.912
Test (Random Over)	0.685	0.358	0.345	0.339
Train (SMOTE)	0.958	0.958	0.959	0.959
Test (SMOTE)	0.786	0.362	0.356	0.358

Random Forest

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.893	0.437	0.772	0.483
Test (None)	0.834	0.355	0.393	0.36
Train (Random Over)	0.913	0.913	0.927	0.912
Test (Random Over)	0.693	0.357	0.345	0.34
Train (SMOTE)	0.959	0.959	0.959	0.959
Test (SMOTE)	0.8	0.36	0.362	0.361

Support Vector Machine

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.883	0.354	0.609	0.352
Test (None)	0.856	0.336	0.361	0.32
Train (Random Over)	0.97	0.97	0.972	0.969
Test (Random Over)	0.809	0.359	0.361	0.359
Train (SMOTE)	0.973	0.973	0.973	0.973
Test (SMOTE)	0.825	0.357	0.362	0.355

Based on the above results, we can see that extracting bag of words features from a particular piece of text could also be effective in evaluating its fluency. Using oversampling techniques has boosted training performance significantly (especially SMOTE), although test performance has deteriorated (significantly in K-nearest neighbours). We can see that test sets tend to have low macro value of precision, recall and f1-score, which highlights the skew in the data.

We can observe that K-nearest neighbours performs baldy in comparison with other approaches tried (especially when oversampling is done), while Support Vector Machine tends to achieve decent test accuracies.

5.4. POS n-gram features

For every piece of text in the data, individual sentences in the text are considered, and POS-tagging is performed such that more grammatical information corresponding to a word is available. This tagging was performed every word in the data, such that the data is now transformed such that every word is viewed as its corresponding POS tag, such as NNP (proper noun, singular), PRP (personal pronoun) etc. Further, an n-gram language model was trained on this transformed data, and was utilized to compute n-gram probability features for every textual sample as done in the baseline (considering K highest and K lowest values as features). These POS features were augmented with the actual n-gram probability features, and various algorithms such as Decision Trees, Random Forest, Support Vector Machine, K-nearest neighbours are used for the classification process. The results achieved on using different sampling techniques (without any oversampling, Random Oversampling, SMOTE) for each algorithm can be viewed below.

K-nearest neighbours

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.875	0.333	0.292	0.311
Test (None)	0.871	0.333	0.29	0.31
Train (Random Over)	0.582	0.582	0.571	0.551
Test (Random Over)	0.354	0.33	0.339	0.235
Train (SMOTE)	0.571	0.571	0.579	0.528
Test (SMOTE)	0.261	0.328	0.337	0.193

Decision Tree

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.893	0.57	0.672	0.61
Test (None)	0.778	0.36	0.354	0.357
Train (Random Over)	0.914	0.914	0.928	0.912
Test (Random Over)	0.69	0.355	0.343	0.338
Train (SMOTE)	0.958	0.958	0.959	0.958
Test (SMOTE)	0.779	0.359	0.352	0.354

Random Forest

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.893	0.456	0.725	0.508
Test (None)	0.833	0.356	0.384	0.359
Train (Random Over)	0.914	0.914	0.927	0.912
Test (Random Over)	0.694	0.355	0.343	0.339
Train (SMOTE)	0.958	0.958	0.959	0.958
Test (SMOTE)	0.8	0.358	0.357	0.357

Support Vector Machine

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.875	0.333	0.292	0.311
Test (None)	0.871	0.333	0.29	0.31
Train (Random Over)	0.343	0.343	0.41	0.237
Test (Random Over)	0.746	0.325	0.337	0.304
Train (SMOTE)	0.347	0.347	0.44	0.24
Test (SMOTE)	0.767	0.333	0.337	0.308

It can be said that the addition of POS n-gram features was useful, as there is an improvement in performance with respect to baseline and related experiments. We can see that the performance in all algorithms was better in cases where no oversampling was done, due to the bias in the data. Although, oversampling has improved training performance significantly (note that the test set performance deteriorated). It also worth observing that the precision and recall in all test sets are very poor, as the macro averages were considered and the results were mostly dominated by a single label.

We can observe that Decision Tree and Random Forest approaches seem to achieve better results in comparison with K-nearest neighbours and SVM.

5.5. RNN Language Model

After all the above parameter tuning experiments, the language model which has been used for calculation of perplexity and relevant probabilistic features, has also been altered. A Recurrent Neural Network is trained to function as a language model, which takes a sequence of words as inputs and outputs the probability distribution for the next word.

The architecture consists of an embedding layer in its first step, where the embedding dimension is configured to be 300. The maximum length for an input sequence is taken as 40, based on observations from the dataset, and appropriate padding and truncation is done wherever

necessary. Further, these embeddings from the input sequence are passed through an RNN layer which consists of 300 neurons / units, which captures the information from the input embeddings, processed in each timestep. The output aggregated vector is then passed through a dense layer with neurons equal to the size of the vocabulary, and softmax is further applied on this generated output to get a probability distribution over the whole vocabulary, for the prediction of the next word in the sequence. This vector is compared with the one-hot encoded ground-truth vector to compute loss (categorical cross-entropy), and perform backpropagation. Adam optimizer with a learning rate of 0.001 is used for the model's training (due to its high efficiency and accuracy).

Keeping the rest of the procedure same (computation of a fixed number of language model based probabilistic features - highest and least, using appropriate padding with median probability), additional metrics such as ROUGE-L, ROUGE-S, SLOR, BLEU, N-gram overlap (which are explained in detail in the earlier section) are also implemented and incorporated into the feature vector for the final classification. Various algorithms such as Decision Trees, Random Forest, Support Vector Machine, K-nearest neighbours are used for the classification process. The results achieved on using different sampling techniques (without any oversampling, Random Oversampling, SMOTE) for each algorithm can be viewed below.

K-nearest neighbours

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.875	0.333	0.292	0.311
Test (None)	0.872	0.333	0.291	0.31
Train (Random Over)	0.628	0.628	0.611	0.603
Test (Random Over)	0.396	0.318	0.333	0.247
Train (SMOTE)	0.515	0.515	0.509	0.507
Test (SMOTE)	0.36	0.317	0.335	0.235

Decision Tree

Type	Accuracy	Recall	Precision	F1-score
Train (None)	1.0	1.0	0.999	0.999
Test (None)	0.773	0.339	0.339	0.339
Train (Random Over)	1.0	1.0	1.0	1.0
Test (Random Over)	0.786	0.362	0.362	0.362
Train (SMOTE)	1.0	1.0	1.0	1.0
Test (SMOTE)	0.68	0.351	0.34	0.332

Random Forest

Type	Accuracy	Recall	Precision	F1-score
Train (None)	1.0	0.999	1.0	0.999
Test (None)	0.859	0.341	0.368	0.332
Train (Random Over)	1.0	1.0	1.0	1.0
Test (Random Over)	0.831	0.347	0.357	0.346
Train (SMOTE)	1.0	1.0	1.0	1.0
Test (SMOTE)	0.801	0.351	0.354	0.352

Support Vector Machine

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.875	0.333	0.292	0.311
Test (None)	0.872	0.333	0.291	0.31
Train (Random Over)	0.334	0.334	0.715	0.168
Test (Random Over)	0.004	0.334	0.501	0.003
Train (SMOTE)	0.334	0.334	0.711	0.169
Test (SMOTE)	0.004	0.334	0.307	0.003

Based on the above results, we can see that there is an improvement in train accuracies of most architectures, due to the capacity of retaining more information (high memory) of the recurrent neural network language model. This has led to overfitting which is evident in the comparison between the train and test accuracies. Random oversampling seemed to be a better choice for oversampling in comparsion with SMOTE, based on the performance metrics. Although, we can observe that oversampling has degraded the performance in most architectures (except Decision Tree). We can also see that test sets tend to have low macro value of precision, recall and f1-score, which highlights the skew in the data.

With respect to the previous experiments, an improvement in performance can be observed for the architectures KNN and Random Forest, and a severe drop in performance for Support Vector machine. Random Forest seemed to achieve the best performance, while decision tree also performed reasonably well. The performance of K-nearest neighbours has improved in comparison with previous set of experiments, but still isn't satisfactory.

These details describe the correlation between the architecture and the data distribution, as there is a lot of variation in the observed metrics.

5.6. LSTM Language Model

Similar to the above approach, a Long Short-Term Memory has also been used for the language model, due to its capability of retaining information across longer sequence lengths. The architecture used is also very similar to the above RNN Language model, which includes an embedding layer of dimension 300 and maximum sequence length of 40, 300 neurons / units in the LSTM layer, and the output is passed through a dense layer with number of neurons equal to the vocabulary size, and eventually a softmax is applied to yield a probability distribution across the vocabulary. Categorical cross-entropy and Adam are the loss function, optimizer respectively.

The rest of the procedure is same as in above case, where feature vector contains fixed number of features along with additionally implemented fluency metrics. Decision Trees, Random Forest, Support Vector Machine, K-nearest neighbours are used for the classification process, and the performance is checked for random oversampling, SMOTE and no data manipulation.

K-nearest neighbours

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.875	0.333	0.292	0.311
Test (None)	0.872	0.333	0.291	0.31
Train (Random Over)	0.614	0.614	0.597	0.587
Test (Random Over)	0.395	0.33	0.344	0.254
Train (SMOTE)	0.495	0.495	0.49	0.484
Test (SMOTE)	0.337	0.339	0.343	0.23

Decision Tree

Type	Accuracy	Recall	Precision	F1-score
Train (None)	1.0	1.0	0.999	0.999
Test (None)	0.772	0.343	0.342	0.342
Train (Random Over)	1.0	1.0	1.0	1.0
Test (Random Over)	0.788	0.361	0.358	0.359
Train (SMOTE)	1.0	1.0	1.0	1.0
Test (SMOTE)	0.706	0.358	0.345	0.341

Random Forest

Type	Accuracy	Recall	Precision	F1-score
Train (None)	1.0	0.999	1.0	0.999
Test (None)	0.857	0.338	0.356	0.328
Train (Random Over)	1.0	1.0	1.0	1.0
Test (Random Over)	0.829	0.347	0.356	0.347
Train (SMOTE)	1.0	1.0	1.0	1.0
Test (SMOTE)	0.794	0.348	0.349	0.348

Support Vector Machine

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.875	0.333	0.292	0.311
Test (None)	0.872	0.333	0.291	0.31
Train (Random Over)	0.334	0.334	0.342	0.167
Test (Random Over)	0.003	0.333	0.001	0.002
Train (SMOTE)	0.334	0.334	0.711	0.167
Test (SMOTE)	0.003	0.333	0.001	0.002

We can observe that the performance in this case is very similar to that of the RNN language model - higher train accuracies due to overfitting, Random oversampling outperforming SMOTE, deterioration of performance on using oversampling (higher accuracy without oversampling due to bias in the data - majority prediction).

Random Forest and Decision Tree approaches have performed well, and the performance of Support Vector Machine has dropped significantly (due to the skewed data distribution).

5.7. Transformer Language Model (BERT)

Due to the increasing trend for transformer architectures in capturing information in an effective manner, BERT (Bidirectional Encoder Representations from Transformers) has been exploited to act as a language model component in this system. Pre-trained BERT tokenizer and Pre-trained BERT Masked Language Model (large uncased) are combined to extract relevant probabilistic and perplexity features.

The rest of the procedure is same as in previous cases, where feature vector contains fixed number of features along with additionally implemented fluency metrics. Decision Trees, Random Forest, Support Vector Machine, K-nearest neighbours are used for the classification process, and the performance is checked for random oversampling, SMOTE and no data manipulation.

K-nearest neighbours

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.875	0.333	0.292	0.311
Test (None)	0.872	0.333	0.291	0.31
Train (Random Over)	0.351	0.351	0.476	0.205
Test (Random Over)	0.029	0.336	0.334	0.034
Train (SMOTE)	0.346	0.346	0.414	0.209
Test (SMOTE)	0.031	0.324	0.327	0.039

Decision Tree

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.995	0.989	0.961	0.974
Test (None)	0.767	0.354	0.349	0.351
Train (Random Over)	0.996	0.996	0.996	0.996
Test (Random Over)	0.776	0.351	0.349	0.35
Train (SMOTE)	0.998	0.998	0.998	0.998
Test (SMOTE)	0.727	0.373	0.348	0.347

Random Forest

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.995	0.956	0.991	0.973
Test (None)	0.86	0.333	0.331	0.317
Train (Random Over)	0.996	0.996	0.996	0.996
Test (Random Over)	0.834	0.334	0.333	0.328
Train (SMOTE)	0.998	0.998	0.998	0.998
Test (SMOTE)	0.795	0.35	0.344	0.341

Support Vector Machine

Type	Accuracy	Recall	Precision	F1-score
Train (None)	0.875	0.333	0.292	0.311
Test (None)	0.872	0.333	0.291	0.31
Train (Random Over)	0.337	0.337	0.232	0.205
Test (Random Over)	0.013	0.329	0.047	0.036
Train (SMOTE)	0.344	0.344	0.249	0.209
Test (SMOTE)	0.013	0.329	0.047	0.036

We can observe that the performance in this case is again very similar to that of the RNN and LSTM language models - higher train accuracies due to overfitting, Random oversampling outperforming SMOTE, deterioration of performance on using oversampling (higher accuracy without oversampling due to bias in the data - majority prediction).

Random Forest and Decision Tree approaches have performed well, and the performance of Support Vector Machine has dropped significantly. There is a severe drop in performance in K-nearest neighbours as well as Support Vector Machines (due to the skewed data distribution).

5.8. Correlation scores

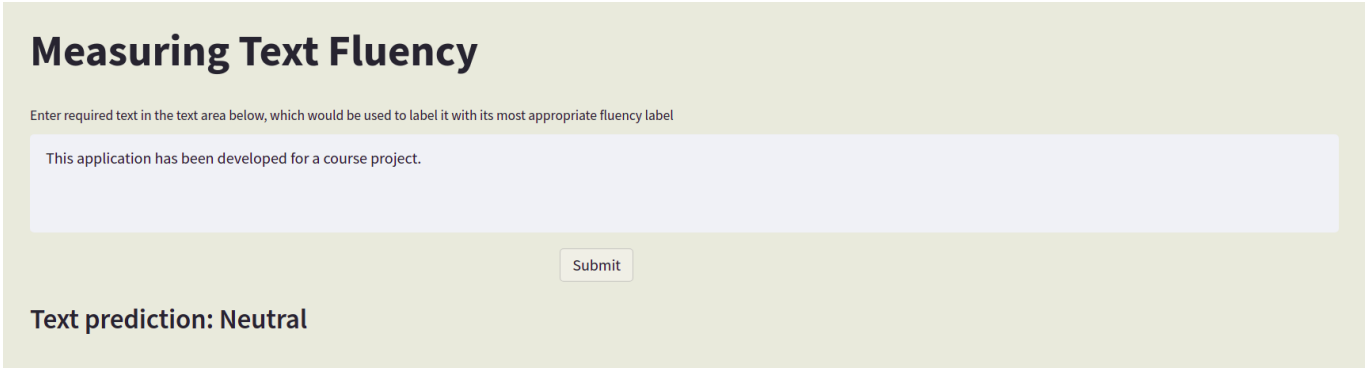
Few features, such as SLOR, ROUGE, N-gram overlap etc. were specifically implemented to compute their correlation with fluency. Their Pearson correlation scores are tabulated below.

Metric	Pearson correlation
SLOR	0.0059
ROUGE-L	0.0608
N-gram overlap	0.0713
BLEU	0.0755
ROUGE-S	**0.0761**

It can be observed that the lowest correlation was obtained by Syntactic Log-odds ratio (SLOR), which is significantly lesser than all other metrics. This points out the fault in relying only on the perplexity and probability values which are generated by the language model. Other metrics such as ROUGE-L, N-gram overlap, BLEU and ROUGE-S take into account the manually annotated references for each data point, and thus, inherently include a human's point of view of fluency and correctness. Thus, higher alignment of the text with these references ideally correspond to a higher value for each of these metrics.

Hence, we can observe that ROUGE-S performs the best, as it considers overlap of subsequences of the reference and the text, preceded by BLEU score and N-gram overlap, which have a similar property. ROUGE-L is also an effective metric, although it has a relatively lesser impact as it focuses only on the length of the longest common subsequence.

We can also observe that all the above 5 metrics are positively correlated with text fluency, and all of them barring SLOR significantly outperform the baseline metrics (n-gram based scoring), confirming their relevance and validity in estimating fluency.



6. Code

The link to the github repository with the complete implementation can be found here.

7. Application

A simple application has been implemented (using the streamlit python library) which takes a piece of text as input and predicts its fluency (Fluent / Neutral / Not Fluent) in real time. The user interface looks as depicted in the image of the dashboard above. On entering the required text and clicking on the Submit button, the corresponding fluency prediction appears below the text box.

For the prediction, the app makes use of the BERT language model with Random Forest classifier (without any oversampling), as this combination has better precision, recall and f1-scores along with accuracy. The models are initially loaded when the app is executed, which takes some time in the beginning, although the subsequent predictions occur in real-time.

The features are extracted in the same manner as done for the sentences in the previous dataset. Although, metrics like ROUGE-L, ROUGE-S, BLEU, N-gram overlap cannot be obtained on feature extraction as they require manual references. Although, the SLOR feature, which is also positively correlated with fluency, is included in the feature vector.

Some examples from the given dataset and their corresponding predictions made by the app are given below.

- **Fluent** - *Kids today face overwhelming pressures, everything from trouble at home to gangs and drugs. Youngsters often find themselves feeling alienated and alone.*
- **Not Fluent** - *It is I, the Great Shopping Avenger, reporting to you from the Great Hall of Consumer Justice, aka the Shopping Avenger's poorly air-conditioned attic office.*
- **Neutral** - *Yet even this angry landowner wound up living happily ever after with the Endangered Species Act. That's because the in- genius Safe Harbor Plan made the conservation worth his while.*

8. Conclusion

Estimating text fluency is becoming an increasingly important task in today's world, due to the large amount of text being generated everyday, and the necessity to maintain formal structure and fluency for better communication. Automating this process is a non-trivial task, which can clearly be visualized in all of the above experiments, where different approaches that correlate highly with the interpretation of fluency, are explored. Obtaining manually annotated dataset is the biggest challenge in generating accurate predictions, as the entire pipeline rests on the training ability of the models in the solution (language models, classifiers etc). There is scope for improvement in different regards such as extracting more features which capture the nuances of the text in a better way, modifying the classification process etc., which can be explored to improve the performance of fluency prediction.

References

[1] Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. Referenceless quality estimation for natural language generation, 2017.

[2] Katharina Kann, Sascha Rothe, and Katja Filippova. Sentence-level fluency evaluation: References help, but can be spared!, 2018.

[3] Chin-Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612, Barcelona, Spain, July 2004.

[4] Ding Liu, Yu Zhou, Chengqing Zong, and Fuji Ren. Automatic evaluation of sentence fluency. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, volume 2, pages 1687–1692 vol.2, 2003.

[5] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[6] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.