# TLMOTE: A Topic-based Language Modelling Approach for Text Oversampling

**Arjun Choudhry, Seba Susan, Anmol Bansal, Anubhav Sharma**
Delhi Technological University, New Delhi, India
{arjunchoudhry_2k18it031, sebasusan, anmolbansal_2k18it025, anubhavsharma_2k18it029}@dtu.ac.in

**Abstract**

Training machine learning and deep learning models on unbalanced datasets can lead to a bias portrayed by the models towards the majority classes. To tackle the problem of bias towards majority classes, researchers have presented various techniques to oversample the minority class data points. Most of the available state-of-the-art oversampling techniques generate artificial data points which cannot be comprehensibly understood by the reader, despite the synthetic data points generated being similar to the original minority class data points. In this work, we present Topic-based Language Modelling Approach for Text Oversampling (TLMOTE), a novel text oversampling technique for supervised learning from unbalanced datasets. TLMOTE improves upon previous approaches by generating data points which can be intelligibly understood by the reader, can relate to the main topics of the minority class, and introduces more variations to the synthetic data generated. We evaluate the efficacy of our approach on various tasks like Suggestion Mining SemEval 2019 Task 9 Subtasks A and B, SMS Spam Detection, and Sentiment Analysis. Experimental results verify that oversampling unbalanced datasets using TLMOTE yields a higher macro F1 score than with other oversampling techniques.

## 1 Introduction

Class imbalance is one of the most prominent issues surrounding supervised learning models for years, despite the significant improvements in classifier architectures and preprocessing approaches. Classifiers trained on unbalanced datasets tend to be biased towards classes with more data points, or the majority classes, reducing the efficacy of the classifier in predicting the correct label for a given input. To this end, significant work has been done to ameliorate the problem of class imbalance in supervised learning tasks. Chawla et al. (2002) presented *Synthetic Minority Oversampling Technique* (SMOTE), a minority class oversampling technique which works in the Euclidean space and has been widely applied to a wide variety of tasks.

Over the years, many variations of SMOTE have been introduced to improve the viability of data points generated for the minority classes. He et al. (2008) introduced *Adaptive Synthetic Sampling Approach for Imbalanced Learning* (ADASYN), a synthetic sampling approach which uses a weighted distribution for the minority classes based on the difficulty level in their learning. Unlike SMOTE, in which each point is given a uniform weight, in ADASYN, a density distribution decides the number of synthetic samples that need to be generated for a specific point. Han, Wang, and Mao (2005) proposed *Borderline-SMOTE*, a variation of SMOTE in which minority class data points whose neighbours are all majority class data points are considered as noise points, and are ignored during oversampling.

Some recent works have also used undersampling of the majority class, or hybrid of oversampling and undersampling approaches for tackling the problem of class imbalance. Susan and Kumar (2018) proposed a hybrid model of minority oversampling and Particle Swarm Optimization-based majority undersampling. These SMOTE-based approaches are applicable across multiple modalities. There have been some approaches proposed to tackle the problem of class imbalance for specific modalities as well. Saini and Susan (2019) proposed a minority oversampling approach for the classification of breast cancer histopathological images, by the generation of synthetic image samples using affine transformations. Moreo, Esuli, and Sebastiani (2016) proposed *Distributed Random Oversampling* (DRO), a text oversampling approach based on assigning a probabilistic generative function to each sample in the minority class of the training set. Anand et al. (2018) tackled class imbalance in Phishing URL Detection datasets using *Text Generative Adversarial Networks*, and generated synthetic URLs similar to those in the original dataset. Leekha, Goswami, and Jain (2020) proposed *Language Model-based Oversampling Technique* (LMOTE), a minority class oversampling technique exclusively for textual data. They found that LMOTE performed comparably to SMOTE, while allowing for a more intuitive understanding of the generated samples.

In this work, we introduce *Topic-based Language Modelling approach for Text Oversampling* (TLMOTE), an oversampling technique using Latent Dirichlet Allocation-based Topic Modelling and Language Modelling for the generation of synthetic data points for the minority classes. We evaluate TLMOTE against SMOTE and LMOTE on Suggestion Mining *SemEval* 2019 Task 9 Subtasks A and B, SMS Spam

Detection, and Sentiment Analysis for a variety of machine learning and deep learning classifiers. The organization of this paper is as follows. TLMOTE is presented in Section 2, the experimental results are discussed in Section 3, and the conclusions are summarized in Section 4.

# 2 Methodology

## 2.1 Datasets and Preprocessing

We use the *SemEval* 2019 Task 9 Subtasks A and B datasets for Suggestion Mining (Negi, Daudert, and Buitelaar, 2019), SMS Spam Classification dataset (Almeida, Hidalgo, and Yamakami, 2011) made available in UCI machine learning repository, and the COVID-19 Vaccine Tweets[1] dataset for sentiment analysis. The Suggestion Mining Subtask A dataset consists of training and testing data from suggestion forums for Windows platform developers, while the Subtask B dataset consists of the same training data as Task A, and testing data from the hotel reviews domain. The Covid Vaccine Tweets dataset consists of tweets on various COVID-19 vaccines, annotated with their correct sentiments. The distributions of these datasets are shown in Table 1.

| Class | Size |
|---|---|
| Sugg | 2085 |
| Non-Sugg | 6415 |

| Class | Size |
|---|---|
| Spam | 747 |
| Ham | 4827 |

| Class | Size |
|---|---|
| Neg | 420 |
| Neut | 3680 |
| Pos | 1900 |

Table 1: Class-wise distribution for Suggestion Mining training dataset, SMS Spam Classification dataset, and COVID-19 Vaccine Tweets dataset for Sentiment Analysis.

The training set for both Suggestion Mining Subtasks A and B are same. During pre-processing, we convert the text to lower case, and de-contract verbs forms (eg. "I'll" to "I will"). We further remove all punctuation marks before training and testing the classifiers on the unbalanced and oversampled data.

## 2.2 Observed Shortcomings of LMOTE

Upon generating synthetic data points using LMOTE for the suggestions class in the suggestion mining train set, we observe that some of the instances generated are highly duplicative. This is especially true for synthetic data points generated using 5-grams with relatively lower frequency.

During the extraction of the most common 5-grams in the suggestions class, the frequency of these 5-grams sharply drops to 1, and we observe 5-grams that occur consecutively within a data point in the initial data. Oversampling using these 5-grams leads to the generation of multiple near-identical data points, which can lead to model over-fitting.

Some examples of successive n-grams generated by the n-gram generator which led to duplicate data points sampled by the language model, along with the generated synthetic samples, are shown in Table 2. We observed many cases of such instances across all datasets oversampled using LMOTE.

[1]https://www.kaggle.com/datasciencetool/covid19-vaccine-tweets-with-sentiment-annotation

| 5-gram | Synthetic datapoint generated by LMOTE |
|---|---|
| i would expect the store | i would expect the store to be capable of searching the partial words also or may be more accessible and the rating can be the only send from the camera using more either h264 or mpeg4 |
| would expect the store to | would expect the store to be capable of searching the partial words also or may be more accessible and the rating can be the only send from the camera using more either h264 or mpeg4 |
| expect the store to be | expect the store to be capable of searching the partial words also or may be more accessible and the rating can be the only send from the camera using more either h264 or mpeg4 |
| the store to be capable | the store to be capable of searching the partial words also or may be more accessible and the rating can be the only send from the camera using more either h264 or mpeg4 |

Table 2: Examples of duplicity in the synthetic data points generated for the suggestion mining dataset using LMOTE, due to consecutively occurring 5-gram seed values.

## 2.3 TLMOTE Proposed Procedure

To reduce the duplicity of generated data points for the minority class, we generate 5-grams which consist of words that relate most to the prominent topics of the minority classes. These words, and thereby the 5-grams are extracted using topic modelling with Latent Dirichlet Allocation (Blei, Ng, and Jordan, 2003).

We further use three language models, which take the last 4 tokens, 5 tokens and 6 tokens, respectively, as the input to predict the next word. We randomly select one word out of the three predictions made by the language models as the next word in the sequence, and append it to the end of the sequence. This process continues till a random number of words between 20 and 30 are appended after the 5-gram. Using three language models randomly, instead of one, introduces more variation into the data points generated, even when similar 5-grams are taken as the seed.

For each minority class of a dataset, TLMOTE follows the following procedure to generate synthetic samples:

**Find top $\alpha$ topics in the minority class** We employ topic modelling using Latent Dirichlet Allocation for extracting the ($\alpha = 100$) most important topics in the minority class samples, using the Gensim package.

**Extract the top 10 keywords $K_t$ for each topic $t$** We further extract the 10 most prominent keywords $K_t$ for each topic $t$ extracted, and create a list $L$ consisting of the set of all top 10 keywords for each topic.

$$L = K_1 \cup K_2 \cup ...K_{100}$$

**Extract the top $\eta$ 5-grams from the minority class samples** We provide $L$ to the n-gram extractor, and extract $\eta$ 5-grams, which contain one or more of the keywords in $L$. $\eta$ equals the deficit of samples in the minority class as compared to the majority class. This ensures that sequences of 5-grams that occur consecutively do not get extracted in significant numbers, as the chain gets broken when a certain 5-gram does not contain a keyword from the list $L$.

**Train three language models on the minority class samples** We train three BiLSTM-based language models on

the minority class samples, which predict the probability distribution of the next word $w_x$ over the entire vocabulary $V$ of the minority class samples, based on the previous 4 words ($w_{x-4}$, $w_{x-3}$, $w_{x-2}$, $w_{x-1}$), 5 words ($w_{x-5}$, $w_{x-4}$, $w_{x-3}$, $w_{x-2}$, $w_{x-1}$) and 6 words ($w_{x-6}$, $w_{x-5}$, $w_{x-4}$, $w_{x-3}$, $w_{x-2}$, $w_{x-1}$), respectively. The language models are trained using GloVe (Pennington, Socher, and Manning, 2014) 100 dimension embeddings. The models predict the probability distributions $P(w_i \mid w_{x-4}\ w_{x-3}\ w_{x-2}\ w_{x-1})\ \forall i \in (V)$, $P(w_i \mid w_{x-5}\ w_{x-4}\ w_{x-3}\ w_{x-2}\ w_{x-1})\ \forall i \in (V)$, and $P(w_i \mid w_{x-6}\ w_{x-5}\ w_{x-4}\ w_{x-3}\ w_{x-2}\ w_{x-1})\ \forall i \in (V)$, such that:

$$w_{x4} = \arg\max_{w_i} P(w_i \mid w_{x-4}\ w_{x-3}\ w_{x-2}\ w_{x-1})$$

$$w_{x5} = \arg\max_{w_i} P(w_i \mid w_{x-5}\ w_{x-4}\ w_{x-3}\ w_{x-2}\ w_{x-1})$$

$$w_{x6} = \arg\max_{w_i} P(w_i \mid w_{x-6}\ w_{x-5}\ w_{x-4}\ w_{x-3}\ w_{x-2}\ w_{x-1})$$

$$w_x = \textbf{random}(w_{x4}, w_{x5}, w_{x6})$$

**Generate synthetic samples using the extracted 5-grams and the three language models** We generate synthetic data points using the three language models and a 5-gram from the set $\eta$, by appending the next word $w_x$ for a random number of times between 20 and 30. This process is repeated for all 5-grams in $\eta$.

The proposed algorithm for TLMOTE is as follows:

**Algorithm 1** Topic-based Language Modelling Approach for Text Oversampling (TLMOTE)

**Input:**
$\mathcal{M}_s = \{r_i \in \mathcal{M} \mid n_i = \{`minority\ class\ sample'\}\}$ Samples from a minority class $\mathcal{M}$;
$t$- Number of topics to extract;
$k$- Number of most prominent keywords per topic;
$\eta$- Number of n-grams to use in TLMOTE;
$n$- type of n-grams *e.g.* 5 for 5-grams, etc.;
$\mathcal{N}$- Number of synthetic samples required.
**Output:** $\mathcal{S} : \mathcal{N}$ synthetic samples for the minority class

$keywords \leftarrow$ **TopicModel**($\mathcal{M}_s$, $t$, $k$)
$topic\_n\_grams \leftarrow$ **NGrams**($\mathcal{M}_s$, $keywords$, $n = 5$, $\mathcal{N}$)
$4gram\_LM \leftarrow$ **TrainLanguageModel**($\mathcal{M}_s$, $n = 4$)
$5gram\_LM \leftarrow$ **TrainLanguageModel**($\mathcal{M}_s$, $n = 5$)
$6gram\_LM \leftarrow$ **TrainLanguageModel**($\mathcal{M}_s$, $n = 6$)
**Initialize** $\mathcal{S} \leftarrow \mathcal{M}_s$
**while** $|\mathcal{S}| < \mathcal{N}$ **do**
  $seed \leftarrow$ **SeedGenerate**($topic\_n\_grams$)
  $sample \leftarrow$ **TLMOTEGenerate**($seed$, $4gram\_LM$, $5gram\_LM$, $6gram\_LM$)
  $\mathcal{S} \leftarrow \mathcal{S} \cup sample$
**end while**
**return** $\mathcal{S}$

Algorithm 1 illustrates our proposed TLMOTE oversampling approach. Some of the procedures used in the algorithm are described below:

**TopicModel**($\mathcal{M}_s$, $t$, $k$): It extracts the top $t$ topics from the minority class samples, and generates the top $k$ keywords by weightage for each topic. It returns the combined list of the most prominent keywords for all $t$ topics.

**NGrams**($\mathcal{M}_s$, $keywords$, $n = 5$, $\eta = \mathcal{N}$): It returns the top $\mathcal{N}$ 5-grams from the minority class samples, which contain one or more *keywords* extracted by **TopicModel**.

**TrainLanguageModel**($\mathcal{M}_s$, $n$): It trains an n-gram BiLSTM-based language model on $\mathcal{M}_s$.

**SeedGenerate**($topic\_based\_n\_grams$): It iterates over the list of *n_grams* provided by **NGrams** and returns the next one as seed for each synthetic sample to be generated.

**TLMOTEGenerate**($seed$, $4gram\_LM$, $5gram\_LM$, $6gram\_LM$): It takes the 4-gram, 5-gram and 6-gram language models, and an n-gram *seed* provided by **SeedGenerate** as input, and generates a synthetic sample for a random length between 25 and 35 tokens as output. Algorithm 2 summarizes the randomised approach used using three language models for the generation of synthetic samples.

We have made the code for TLMOTE available online[2].

**Algorithm 2** TLMOTEGenerate procedure from TLMOTE algorithm

**Input:**
$seed$ - n-gram to be used for generating synthetic sample;
$4gram\_LM$- 4-gram Language Model;
$5gram\_LM$- 5-gram Language Model;
$6gram\_LM$- 6-gram Language Model;
$t$- Number of tokens used as input to the language models;
$min$- Minimum length of tokens to be appended to seed in synthetic sample;
$max$- Maximum length of tokens to be appended to seed in synthetic sample;
**Output:** $Sent : Synthetic\ sample\ generated$

$length \leftarrow$ **random**($min$, $max$)
**Initialize** *Sent* $\leftarrow$ *seed*
**while** |*Sent*| < *length* **do**
  **Initialize** *words* $\leftarrow$ [ ]
  *words* $\leftarrow$ *words* $\cup$ $4gram\_LM$(**Last_t_tokens**($Sent$, $t = 4$))
  *words* $\leftarrow$ *words* $\cup$ $5gram\_LM$(**Last_t_tokens**($Sent$, $t = 5$))
  **if** |*Sent*| >= 6 **then**
    *words* $\leftarrow$ *words* $\cup$ $6gram\_LM$(**Last_t_tokens**($Sent$, $t = 6$))
  **end if**
  $next\_token \leftarrow$ **RandomWord**($words$)
  $Sent \leftarrow$ **Concatenate**($Sent$, $next\_token$)
**end while**
**return** $Sent$

Some of the procedures used in Algorithm 2 are described below:

[2]https://github.com/Arjun7m/TLMOTE

**random**($min,\ max$): It generates a number between *min* and *max*.

**Last_t_tokens**($Sent,\ t$): It returns the last $t$ tokens in the list *Sent*.

**RandomWord**($words$): It returns one of the tokens in the list $words$ randomly.

**Concatenate**($Sent,\ next\_token$): It appends the generated *next_token* to the end of *Sent*.

### 2.4 Classification Models

We evaluate the efficacy of TLMOTE by training and evaluating a variety of machine learning and deep learning classifiers on the unbalanced datasets, as well as datasets oversampled using SMOTE, LMOTE, TLMOTE and one of its variations. Among machine learning classifiers, we train and evaluate Random Forest (RF), Support Vector Machine (SVM) and Logistic Regression (LR) classifiers, while among deep learning architectures, we train and evaluate CNN-LSTM (Zhou et al., 2015) neural network, Attention-based Convolutional Bi-LSTM neural network (ACBiLSTM) (Kamyab, Liu, and Adjeisah, 2021) and Attention-based Recurrent Convolutional neural network (ARC) (Guo et al., 2019).

We optimise all machine learning models using GridSearch, while all deep learning models are optimised using Early Stopping to prevent over-fitting.

## 3 Results and Discussion

The experiments were performed using Python 3.7 version software on an NVIDIA Tesla P100 GPU. We evaluated the performance of a variety of deep learning and machine learning classifiers trained on the unbalanced datasets, as well as datasets oversampled using SMOTE, LMOTE and TLMOTE (referred to as TLMOTE (R) in the tables). To assess the impact of using three language models in a randomised order, we also oversample the datasets using a variation of TLMOTE (referred to as TLMOTE (M) in the tables), which selects the token with the highest probability among the three outputs of the language models, instead of choosing randomly.

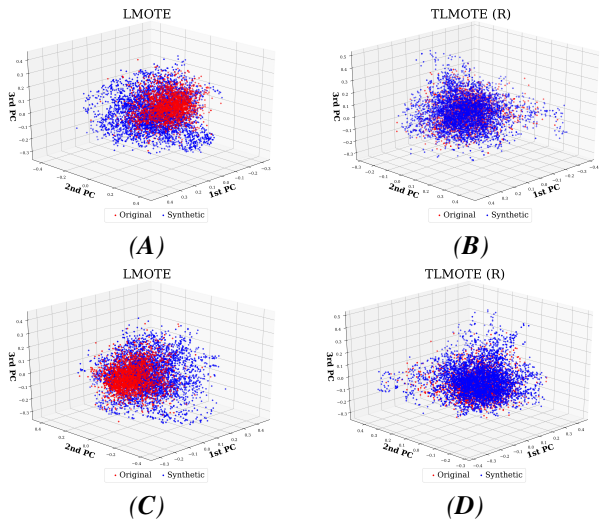


Figure 1: 3-dimensional PCA plots for the Suggestion Mining tasks oversampled using: (A) LMOTE (B) TLMOTE (C) LMOTE Reverse View (D) TLMOTE Reverse View.

| Model | Class | Unbalanced | SMOTE | LMOTE | TLMOTE(M) | TLMOTE(R) |
|---|---|---|---|---|---|---|
| | | F1 | F1 | F1 | F1 | F1 |
| SVM | Non-suggestion | 0.91 | 0.88 | 0.90 | 0.91 | **0.92** |
| | Suggestion | 0.33 | 0.38 | 0.36 | **0.38** | **0.38** |
| | Macro avg | 0.62 | 0.63 | 0.63 | 0.64 | **0.65** |
| RF | Non-suggestion | **0.95** | 0.91 | 0.94 | 0.94 | 0.94 |
| | Suggestion | 0.32 | 0.46 | 0.48 | **0.51** | 0.50 |
| | Macro avg | 0.64 | 0.69 | 0.71 | **0.72** | **0.72** |
| LR | Non-suggestion | 0.88 | 0.87 | 0.88 | 0.89 | **0.90** |
| | Suggestion | 0.29 | 0.31 | 0.33 | 0.36 | **0.38** |
| | Macro avg | 0.58 | 0.59 | 0.61 | 0.63 | **0.64** |

Table 3: Performance evaluation of various Machine Learning models trained on the SemEval 2019 Task 9 Subtask A.

| Model | Class | Unbalanced | LMOTE | TLMOTE(M) | TLMOTE(R) |
|---|---|---|---|---|---|
| | | F1 | F1 | F1 | F1 |
| CNN-LSTM | Non-suggestion | 0.92 | 0.94 | 0.93 | **0.95** |
| | Suggestion | 0.49 | 0.51 | 0.55 | **0.58** |
| | Macro avg | 0.70 | 0.72 | 0.74 | **0.76** |
| ACBiLSTM | Non-suggestion | 0.93 | **0.96** | 0.94 | 0.95 |
| | Suggestion | 0.56 | 0.58 | 0.61 | **0.62** |
| | Macro avg | 0.75 | 0.77 | 0.78 | **0.79** |
| ARC | Non-suggestion | **0.95** | 0.94 | **0.95** | **0.95** |
| | Suggestion | 0.54 | 0.55 | 0.56 | **0.58** |
| | Macro avg | 0.74 | 0.75 | 0.75 | **0.77** |

Table 4: Performance evaluation of various Deep Learning models trained on the SemEval 2019 Task 9 Subtask A.

| Model | Class | Unbalanced | SMOTE | LMOTE | TLMOTE(M) | TLMOTE(R) |
|---|---|---|---|---|---|---|
| | | F1 | F1 | F1 | F1 | F1 |
| SVM | Non-suggestion | **0.73** | **0.73** | 0.72 | 0.72 | 0.71 |
| | Suggestion | 0.06 | **0.21** | 0.08 | 0.12 | 0.17 |
| | Macro avg | 0.39 | **0.47** | 0.40 | 0.42 | 0.44 |
| RF | Non-suggestion | **0.74** | **0.74** | **0.74** | **0.74** | 0.73 |
| | Suggestion | 0.04 | 0.10 | 0.10 | 0.10 | **0.12** |
| | Macro avg | 0.39 | 0.42 | 0.42 | 0.42 | **0.43** |
| LR | Non-suggestion | 0.71 | 0.71 | 0.71 | **0.72** | 0.71 |
| | Suggestion | 0.22 | **0.25** | 0.23 | 0.23 | **0.25** |
| | Macro avg | 0.46 | **0.48** | 0.47 | **0.48** | **0.48** |

Table 5: Performance evaluation of various Machine Learning models trained on the SemEval 2019 Task 9 Subtask B.

We use the provided train-test split for the Suggestion Mining tasks, and randomly split the other datasets in the ratio 80:20 for training and testing sets of the other datasets. For the sake of a fair comparison, we use the same training splits for all oversampling techniques. Performance evaluations for all combinations of classifiers and oversampling techniques using F1 score for each class, as well as Macro F1 score for the total testing set, are illustrated in Tables 3 and 4 for Suggestion Mining Subtask A, Tables 5 and 7 for Suggestion Mining Subtask B, Tables 8 and 9 for the SMS Spam Detection task, and Tables 10 and 11 for the Sentiment Analysis task. Figure 1 represents Principal Component Analysis (PCA) (Jolliffe, 2002) graphs for the oversampled training sets generated for the suggestion mining task using TLMOTE (R) and LMOTE, plotted using text embeddings generated using a Tf-IDf vectorizer fit on the original data points. Some prominent findings we observed are:

**Classifiers trained on datasets oversampled using TLMOTE outperformed those trained on datasets oversam-**

| 5-gram | Synthetic datapoint generated by TLMOTE |
|---|---|
| please make support for setting | please make support for setting set automationid on corewindows and messagedialog could be supported in windows 10 too popular apps are displayed to every phone |
| make support for setting set | make support for setting set automationid on corewindows and messagedialog should be added to uwp for wp images should be tailored to be in a specific way |
| support for setting set automationid | support for setting set automationid on corewindows and messagedialog should be supported in store apps like we have download media like for windows 78 for windows 8 plz see this |

Table 6: Examples of synthetic data points generated using TLMOTE for the suggestion mining training set. Data points show variations, even for consecutively occurring 5-gram seed values.

| Model | Class | Unbalanced | LMOTE | TLMOTE(M) | TLMOTE(R) |
|---|---|---|---|---|---|
| | | F1 | F1 | F1 | F1 |
| CNN-LSTM | Non-suggestion | 0.74 | **0.75** | **0.75** | 0.74 |
| | Suggestion | 0.25 | 0.27 | 0.33 | **0.34** |
| | Macro avg | 0.49 | 0.51 | **0.54** | **0.54** |
| ACBiLSTM | Non-suggestion | **0.75** | **0.75** | **0.75** | **0.75** |
| | Suggestion | 0.29 | 0.24 | **0.32** | **0.32** |
| | Macro avg | 0.52 | 0.50 | **0.54** | 0.53 |
| ARC | Non-suggestion | **0.75** | 0.74 | **0.75** | **0.75** |
| | Suggestion | 0.26 | 0.20 | 0.26 | **0.28** |
| | Macro avg | 0.50 | 0.47 | 0.50 | **0.52** |

Table 7: Performance evaluation of various Deep Learning models trained on the SemEval 2019 Task 9 Subtask B.

| Model | Class | Unbalanced | SMOTE | LMOTE | TLMOTE(M) | TLMOTE(R) |
|---|---|---|---|---|---|---|
| | | F1 | F1 | F1 | F1 | F1 |
| SVM | Ham | 0.98 | **0.99** | **0.99** | **0.99** | **0.99** |
| | Spam | 0.88 | 0.96 | 0.94 | 0.95 | **0.97** |
| | Macro avg | 0.93 | **0.98** | 0.96 | 0.97 | **0.98** |
| RF | Ham | **0.99** | **0.99** | **0.99** | **0.99** | **0.99** |
| | Spam | 0.90 | 0.92 | 0.94 | **0.96** | **0.96** |
| | Macro avg | 0.94 | 0.95 | 0.96 | **0.98** | **0.98** |
| LR | Ham | 0.98 | **0.99** | **0.99** | **0.99** | **0.99** |
| | Spam | 0.88 | 0.95 | 0.95 | 0.96 | **0.97** |
| | Macro avg | 0.93 | 0.97 | 0.97 | **0.98** | **0.98** |

Table 8: Performance evaluation of various Machine Learning models trained on the SMS Spam Filtering dataset.

**pled using SMOTE and LMOTE** We observed that classifiers trained using TLMOTE outperformed those trained using SMOTE and LMOTE in terms of the macro F1 score across nearly all datasets and model architectures. In most cases, we observed that classifiers trained on datasets oversampled using TLMOTE portrayed a 1-5 percent improvement in macro F1 score over their counterparts trained on datasets oversampled using SMOTE and LMOTE, indicating significantly reduced bias shown by the classifiers trained on datasets oversampled using TLMOTE towards the majority classes. We saw slightly greater improvements with TLMOTE, as compared to LMOTE, in case of deep learning models (Tables 4, 7, 9 and 11) when compared with machine learning models (Tables 3, 5, 8 and 10) on the same datasets. This can be attributed to over-fitting in case of deep learning models when trained on datasets oversampled using LMOTE. Due to increased variations in the instances generated by TLMOTE, deep learning models trained on datasets oversampled using TLMOTE avoid the issue of overfitting.

**Oversampling with three language models in a randomized order leads to marginal improvements in the performance of the classifiers** We observed that classifiers trained using TLMOTE with three language models, with 4, 5 and 6 tokens as input length, respectively, and used in a randomized order (referred to as TLMOTE (R) in the tables), outperform their counterparts trained on datasets oversampled using TLMOTE (M), where the next word is taken from that language model which gives the highest probability for a word in the probability distribution.

| Model | Class | Unbalanced | LMOTE | TLMOTE(M) | TLMOTE(R) |
|---|---|---|---|---|---|
| | | F1 | F1 | F1 | F1 |
| CNN-LSTM | Ham | 0.99 | **1.00** | **1.00** | **1.00** |
| | Spam | 0.95 | 0.97 | **0.98** | **0.98** |
| | Macro avg | 0.97 | 0.98 | **0.99** | **0.99** |
| ACBiLSTM | Ham | 0.99 | **1.00** | 0.99 | **1.00** |
| | Spam | 0.96 | **0.98** | 0.97 | **0.98** |
| | Macro avg | 0.98 | **0.99** | 0.98 | **0.99** |
| ARC | Ham | 0.99 | 0.99 | 0.99 | **1.00** |
| | Spam | 0.95 | **0.97** | **0.97** | **0.97** |
| | Macro avg | 0.97 | **0.98** | **0.98** | **0.98** |

Table 9: Performance evaluation of various Deep Learning models trained on the SMS Spam Filtering dataset.

| Model | Class | Unbalanced | SMOTE | LMOTE | TLMOTE(M) | TLMOTE(R) |
|---|---|---|---|---|---|---|
| | | F1 | F1 | F1 | F1 | F1 |
| SVM | Negative | 0.00 | **0.33** | 0.31 | 0.32 | 0.32 |
| | Neutral | **0.82** | 0.73 | 0.76 | 0.77 | 0.75 |
| | Positive | 0.57 | 0.62 | 0.61 | 0.60 | **0.63** |
| | Macro avg | 0.46 | 0.56 | 0.56 | **0.57** | **0.57** |
| RF | Negative | 0.00 | 0.18 | 0.17 | **0.28** | **0.28** |
| | Neutral | **0.79** | **0.79** | 0.78 | 0.76 | **0.79** |
| | Positive | 0.51 | **0.61** | 0.57 | 0.59 | 0.60 |
| | Macro avg | 0.44 | 0.53 | 0.51 | 0.55 | **0.56** |
| LR | Negative | 0.08 | 0.34 | **0.36** | 0.33 | **0.36** |
| | Neutral | **0.81** | 0.76 | 0.79 | 0.77 | 0.78 |
| | Positive | 0.60 | **0.64** | 0.62 | 0.63 | 0.60 |
| | Macro avg | 0.49 | 0.58 | **0.59** | 0.57 | 0.58 |

Table 10: Performance evaluation of various Machine Learning models trained on the Sentiment Analysis dataset.

| Model | Class | Unbalanced | LMOTE | TLMOTE(M) | TLMOTE(R) |
|---|---|---|---|---|---|
| | | F1 | F1 | F1 | F1 |
| CNN-LSTM | Negative | 0.19 | 0.26 | **0.31** | 0.28 |
| | Neutral | **0.78** | 0.75 | 0.77 | 0.77 |
| | Positive | 0.59 | **0.60** | 0.59 | **0.60** |
| | Macro avg | 0.52 | 0.54 | **0.56** | 0.55 |
| ACBiLSTM | Negative | 0.11 | 0.20 | 0.23 | **0.25** |
| | Neutral | 0.77 | 0.77 | **0.78** | **0.78** |
| | Positive | 0.55 | **0.56** | 0.54 | **0.56** |
| | Macro avg | 0.48 | 0.51 | 0.52 | **0.53** |
| ARC | Negative | 0.21 | **0.30** | 0.23 | 0.28 |
| | Neutral | 0.75 | 0.71 | **0.79** | 0.77 |
| | Positive | 0.59 | 0.59 | 0.59 | **0.63** |
| | Macro avg | 0.52 | 0.53 | 0.54 | **0.56** |

Table 11: Performance evaluation of various Deep Learning models trained on the Sentiment Analysis dataset.

**Upon dimensionality reduction using Principal Component Analysis, synthetic samples generated using TLMOTE more closely and uniformly resemble the original samples that those generated using LMOTE** We observed that synthetic samples from TLMOTE more closely follow the boundary of the original samples in the PCA plots in Fig-

ure 1, as compared to those generated using LMOTE, which are skewed further away from the original data points. Further, the distribution is more uniform in case of TLMOTE samples.

**Synthetic datapoints generated using TLMOTE (R) show significantly lower duplicity than those generated using LMOTE for consecutively occurring 5-gram seeds** We observed that synthetic samples generated using TLMOTE (R) in Table 6 show substantial variations even for consecutively occurring 5-grams extracted by the topic-based n-gram generator. This reduces the possibility of the classifiers to overfit on the training set, and thus help improve performance on the test set.

## 4 Conclusion

In this work, we presented Topic-based Language Modelling approach for Text Oversampling (TLMOTE), a variation of LMOTE with three language models with 4, 5 and 6 tokens as input, respectively, used in a randomized order, and 5-gram seeds extracted based on the words relating to the most important topics in the minority class data points. Experimental evaluations show that classifiers trained on datasets oversampled with TLMOTE outperform those trained on datasets oversampled using other approaches like SMOTE and LMOTE in terms of minority class F1 score and macro-averaged F1 score.

## References


Almeida, T. A.; Hidalgo, J. M. G.; and Yamakami, A. 2011. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 11th ACM Symposium on Document Engineering*, DocEng '11, 259–262. New York, NY, USA: Association for Computing Machinery.

Anand, A.; Gorde, K.; Antony Moniz, J. R.; Park, N.; Chakraborty, T.; and Chu, B.-T. 2018. Phishing url detection with oversampling based on text generative adversarial networks. In *2018 IEEE International Conference on Big Data (Big Data)*, 1168–1177.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.* 3(null):993–1022.

Chawla, N.; Bowyer, K.; Hall, L.; and Kegelmeyer, W. 2002. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)* 16:321–357.

Guo, X.; Zhang, H.; Yang, H.; Xu, L.; and Ye, Z. 2019. A single attention-based combination of cnn and rnn for relation classification. *IEEE Access* 7:12467–12475.

Han, H.; Wang, W.-Y.; and Mao, B.-H. 2005. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In Huang, D.-S.; Zhang, X.-P.; and Huang, G.-B., eds., *Advances in Intelligent Computing*, 878–887. Berlin, Heidelberg: Springer Berlin Heidelberg.

He, H.; Bai, Y.; Garcia, E. A.; and Li, S. 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 1322–1328.

Jolliffe, I. 2002. *Principal component analysis*. New York: Springer Verlag.

Kamyab, A.; Liu, G.; and Adjeisah, M. 2021. Attention-based cnn and bi-lstm model based on tf-idf and glove word embedding for sentiment analysis. *Applied Sciences*.

Leekha, M.; Goswami, M.; and Jain, M. 2020. A multi-task approach to open domain suggestion mining using language model for text over-sampling. In *Lecture Notes in Computer Science*. Springer International Publishing. 223–229.

Moreo, A.; Esuli, A.; and Sebastiani, F. 2016. Distributional random oversampling for imbalanced text classification. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, 805–808. New York, NY, USA: Association for Computing Machinery.

Negi, S.; Daudert, T.; and Buitelaar, P. 2019. SemEval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, 877–887. Minneapolis, Minnesota, USA: Association for Computational Linguistics.

Pennington, J.; Socher, R.; and Manning, C. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. Doha, Qatar: Association for Computational Linguistics.

Saini, M., and Susan, S. 2019. Data augmentation of minority class with transfer learning for classification of imbalanced breast cancer dataset using inception-v3. In *Pattern Recognition and Image Analysis: 9th Iberian Conference, IbPRIA 2019, Madrid, Spain, July 1–4, 2019, Proceedings, Part I*, 409–420. Berlin, Heidelberg: Springer-Verlag.

Susan, S., and Kumar, A. 2018. Hybrid of intelligent minority oversampling and pso-based intelligent majority undersampling for learning from imbalanced datasets. In Abraham, A.; Cherukuri, A. K.; Melin, P.; and Gandhi, N., eds., *Intelligent Systems Design and Applications - 18th International Conference on Intelligent Systems Design and Applications, ISDA 2018, Vellore, India, December 6-8, 2018, Volume 2*, volume 941 of *Advances in Intelligent Systems and Computing*, 760–769. Springer.

Zhou, C.; Sun, C.; Liu, Z.; and Lau, F. C. M. 2015. A c-lstm neural network for text classification.