SMS Classifier : Ham or Spam

Abhishek Ranjan
Computer Science
San Diego State University
San Diego, California

Anuj Kawane
Computer Science
San Diego State University
San Diego, California

***Abstract*—In recent years, We've all started receiving many spam SMS every day. The cost of sending a text message has dropped significantly in the last decade, and sending an SMS is now free in most cases. Unfortunately, spammers have used this to transmit millions of spam messages or unwanted commercial advertisements (spam) every day. An effective spam detection is a critical tool for assisting individuals in determining whether or not an SMS is a spam. In this report, we used a combination of word embedding technique, re-sampling, ensemble and machine learning techniques to detect SMS spam. We built a spam detection model using a dataset from UCI.**

I. Introduction

Mobile phones have become one of many people's attached companions. With the rapid use of mobile devices and millions of people sending messages on a daily basis, However, the increase in mobile users and low-cost SMS text messages has the unintended consequence of drawing more unsolicited bulk communications, particularly in the form of advertisements.

SMS is a text communication platform that allows users to exchange short text messages of fixed length (160 characters) utilizing established communication protocols through mobile devices. Spam, on the other hand, is defined as "unwanted, unsolicited digital communication sent in bulk". These spam are typically delivered by businesses looking to market and promote their products or services to recipients. Besides promoting materials, spams also can threaten users' privacy with phishing, fraud and identity theft attacks through text messages.

The goal of this project is to use machine learning techniques to distinguish between spam and legitimate messages. To make the procedure more agile and efficient, machine learning and Natural Language Processing techniques were applied.

In this Project, we have outlined multiple goals.

- To explore and compare various word-embedding techniques such as TF-IDF, GloVe Embeddings(used with LSTM model), BertTokenizer(with BERT models)
- To study the effects of re-sampling of training sets like SMOTE.
- To explore the effect of various ensemble methods like XGBoost and random forest.
- To run various machine learning models like NaiveBayes, KNN, Support Vector Machines, Logistic Regression and assess its performance.

This project report is organized as follows:

- Datasets, Initial Analysis, Text Cleaning
- Baseline Machine Learning model with Bag of Words
- Ensemble Methods
- Re-sampling strategies
- GloVe Embeddings with LSTM
- BERT model
- Conclusion

II. Datasets, Initial Analysis, Text Cleaning

Dataset contained the csv file with 5572 entries. We dropped three columns that were significantly missing leaving us with two columns text and target. Text contains the SMS and Target signifies if the message is Ham or Spam. The distribution between targets is shown below table.

TABLE I
Distribution of Dataset

Target	Count
HAM	4875
SPAM	747

Fig. 1. Steps to transform original message

Based on Initial Data analysis, we found that Ham messages averaged around 10 words. On the other hand, Spam averaged at 30 words. Similar trends are noticed with respect to message

length. Spam averaged 150 length whereas Ham averaged around 45. We removed all non-alphabetic characters with a space, converted all characters to lowercase. Post which tokenization and Lemmatization is performed. We also removed all the stop words from the text.

III. Baseline Machine Learning model with Bag of Words

In this section, we have explored the various Machine Learning models trained and their performance with Bag of Words technique. We used Tf-IDF as a word encoding technique. tf–idf short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

The multinomial Naive Bayes classifier is used as a baseline as it is suitable for classification with discrete features. With 75% of the data in the training set, the model had 96.64% as a CV score. It achieved 95.90% accuracy on the test set.

KNN classifiers have also been trained and evaluated. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. It performed decently with 91.12% as CV score and performance on testing set as 91.6%.

We also tested C-Support Vector Classification based on libsvm using the rbf kernel and degree as 3. It resulted in a mean CV score of 97% and performance on testing set as 97.5%.

With L2 penalty and fit_intercept, Logistic Regression performed decently with 96.05% as a validation score and testing performance as 94.75%.

Decision Tree classifier with unlimited max depth and quality of split chosen as 'gini' for Gini impurity. It yielded a validation score of 95.97% and testing performance of 96.12%.

IV. Ensemble Methods

Ensemble methods consist of training numerous prediction models for the same prediction task and integrating their outputs to generate the final prediction [1] [2] [3] . Because aggregating the predictions from numerous models usually reduces the overfitting phenomena, ensembles of models often provide better prediction performance than single models.

The two types of ensemble methods are:

- Parallel-based
- Iterative-based ensembles.

Each baseline learner is trained in parallel in parallel-based ensembles, utilizing either a subset of the training data, a subset of the training features, or a mix of both. The baseline classifiers are trained in sequence in iterative-based ensembles, often known as boosting, with each learner in the sequence seeking to minimize the prediction errors of the prior learner. Bagging and random forest are the two most prevalent techniques for parallel-based ensembles. In iterative-based ensembles. The currently most widely-used implementations for boosting are XGBoost [4], CatBoost [5] and LightGBM [6].

We have tested Random Forest in Iterative-based ensembles and XGBoost in parallel-based ensembles. For Random forest, with the number of trees in the forest to be 100 and unlimited max-depth yielded 95.97% and a test score of 96.12%. XGBoost on the other hand with the default parameter with number of jobs as 1 yielded validation score 96.9% and test set accuracy of 95.40%.

V. Re-sampling strategies

TABLE II
Model Performance

Model	Precision	Recall	Test Accuracy	Train Accuracy
NaiveBayes	0.992647	0.706806	0.959081	0.976071
RandomForest	1.000000	0.811518	0.974156	1.000000
KNeighbours	1.000000	0.392670	0.916726	0.925580
SVC	0.987261	0.811518	0.972721	0.997846
Logistic Regression	0.960938	0.643979	0.947595	0.973199
Decision Tree	0.891429	0.816754	0.961235	1.000000
XGBoost	0.970370	0.685864	0.954056	0.976549

As we know, the number of ham is much higher than the number of spams. And if we look at the recall in above table, it's quite low. This simply suggests that the fraction of true positives correctly detected is low. So we decided to try a re-sampling method and see how it affects the model's performance.

By re-sampling the dataset to minimize the imbalance ratio, re-sampling procedures address class imbalance at the data level. Before training the prediction model, an imbalanced dataset is re-sampled, which can be thought of as a data pretreatment step. For re-sampling imbalanced datasets, a variety of methods have been developed, which can be divided into three categories: oversampling, undersampling, and hybrid strategies.

TABLE III
Model Performance in Re-sampled Data

Model	Precision	Recall	Test Accuracy	Train Accuracy
NaiveBayes	0.852381	0.937173	0.969131	0.987441
RandomForest	1.000000	0.816754	0.974874	1.000000
KNeighbours	0.977444	0.680628	0.954056	0.998344
SVC	0.993671	0.821990	0.974874	1.000000
Logistic Regression	0.937143	0.858639	0.972721	0.991306
Decision Tree	0.795122	0.853403	0.949749	1.000000
XGBoost	0.796954	0.821990	0.946877	0.978747

SMOTE generates synthetic cases in the vicinity of observed ones to oversample the minority class. The objective is to interpolate across samples of the same class to create new minority examples. This results in clusters forming around each minority observation. The classifier creates larger decision areas with neighboring instances from the minority class by constructing synthetic observations. In numerous instances, SMOTE has been proven to improve the performance of a base classifier [7] [8].

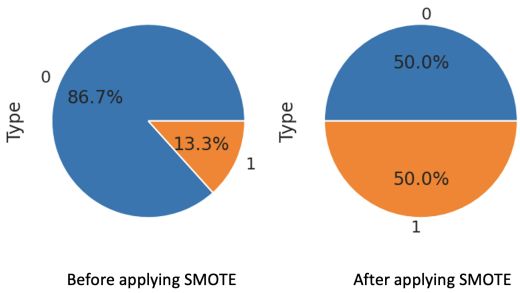

Fig. 2. Distribution of data in training sample

TABLE IV
VALIDATION SCORE ON ORIGINAL TRAINING SET

Model	Count
NaiveBayes	0.966497
RandomForest	0.977744
KNeighbours	0.911942
SVC	0.975829
Logistic Regression	0.960514
Decision Tree	0.959799
XGBoost	0.969130

TABLE V
VALIDATION SCORE ON RE-SAMPLED TRAINING SET

Model	Count
NaiveBayes	0.979852
RandomForest	0.994344
KNeighbours	0.994204
SVC	0.997793
Logistic Regression	0.986478
Decision Tree	0.976404
XGBoost	0.972955

We can observe from the above tables that recall, testing set accuracy (refer table 2 and 3) and validation score (refer table 4 and 5) significantly improved. The most notable improvement in test performance can be observed in KNN. The testing performance shoots up from 91.6% to 95.4%. The improvement in Recall score is pretty loud across all the machine learning models. Most notably, the Recall for Naive Bayes jumped from 70.68% to 93.71%.

VI. GloVe Embeddings with LSTM

GloVe is an unsupervised learning technique that generates word vector representations. The resulting representations highlight interesting linear substructures of the word vector space, which are trained using aggregated global word-word co-occurrence information from a corpus. They are a form of word representation that try to merge human understanding of languages into their structure. They have a learned representation in an n-dimensional space, where words with similar meanings have similar embeddings. Two similar words are represented by almost similar vectors that are at a small distance in the vector space.

Bi-directional LSTM was used to train the embedded matrix. Long Short-Term Memory (LSTM) is a model that was first proposed in 1997. Bidirectional LSTM is simply an extension of the Gated Recurrent Neural Network model. The crucial aspect is that these networks may retain data for future cell processing. We can think of LSTM as an RNN with two key vectors and a memory pool.

- Short-term state: keeps the output at the current time step.
- Long-term state: stores, reads, and rejects items meant for the long-term while passing through the network.

The decision of reading, storing, and writing is based on some activation functions. The output from those activate functions is a value between (0, 1).

In bidirectional LSTM, instead of training a single model, we introduce two. The first model learns the sequence of the input provided, and the second model learns the reverse of that sequence.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 77, 100)           659700

bidirectional (Bidirectiona  (None, 77, 154)           109648
l)

global_max_pooling1d (Globa  (None, 154)               0
lMaxPooling1D)

batch_normalization (BatchN  (None, 154)               616
ormalization)

dropout (Dropout)            (None, 154)               0

dense (Dense)                (None, 77)                11935

dropout_1 (Dropout)          (None, 77)                0

dense_1 (Dense)              (None, 77)                6006

dropout_2 (Dropout)          (None, 77)                0

dense_2 (Dense)              (None, 1)                 78

=================================================================
Total params: 787,983
Trainable params: 787,675
Non-trainable params: 308
_________________________________________________________________
```

Fig. 3. Model Summary for bidirectional LSTM with GLoVE embedding

We trained the bidirectional LSTM with epoch 10 and observed the validation accuracy of 98.35% and 98.34% on testing results as well.

TABLE VI
CONFUSION MATRIX

		Predicted Classes	
		Ham	Spam
Actual Classes	Ham	1195	10
	Spam	13	175

VII. BERT Model

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning

community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 77)]	0	[]
input_2 (InputLayer)	[(None, 77)]	0	[]
tf_bert_model (TFBertModel)	TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 77, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)	109482240	['input_1[0][0]', 'input_2[0][0]']
dense_3 (Dense)	(None, 32)	24608	['tf_bert_model[0][1]']
dropout_40 (Dropout)	(None, 32)	0	['dense_3[0][0]']
dense_4 (Dense)	(None, 1)	33	['dropout_40[0][0]']

Total params: 109,506,881
Trainable params: 109,506,881
Non-trainable params: 0

Fig. 4. Model Summary for BERT

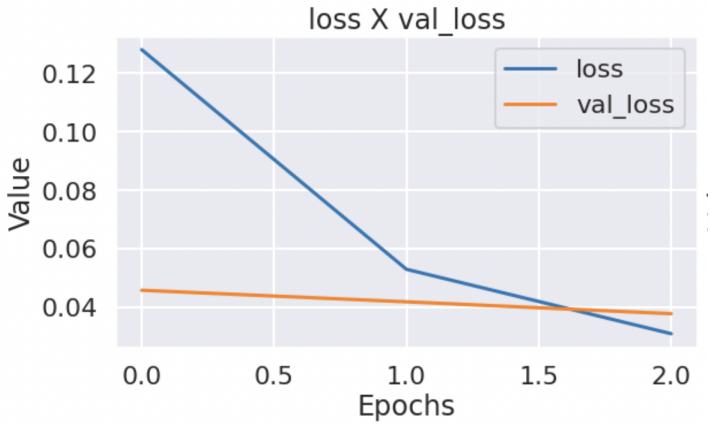

Fig. 5. Loss vs Epochs

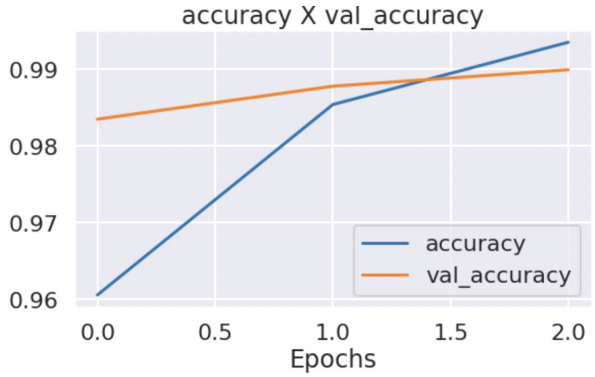

Fig. 6. Accuracy vs Epochs

BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible. One of the major reasons for the success of BERT is that it is a context-based embedding model, unlike other popular embedding models, such as word2vec, which are context-free.

We trained using pre-trained BERT(TFBertModel) with epoch of 3 and batch size of 10 and found validation accuracy of 99%.

VIII. Conclusion

We have studied the impact of re-sampling in performance of various Machine Learning algorithms. It significantly improved testing and validation accuracy, recall for all the ML algorithms. After re-sampling the data, Random Forest, KNN, SVC showed a validation score of around 99%. Along with it, the BERT model and LSTM with Glove embeddings also showed excellent performance. For further improvement in detecting spams, we will need the updated data as the current data is really outdated and may not reflect the current scenario.

References

[1] Gianluca Bontempi. Statistical foundations of machine learning, 2nd edition. Université Libre de Bruxelles, 2021.

[2] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. Learning from imbalanced data sets. Springer, 2018.

[3] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: review of methods and applications. Expert Systems with Applications, 73:220–239, 2017.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: a scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 785–794. 2016.

[5] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. arXiv preprint arXiv:1810.11363, 2018

[6] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: a highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30:3146–3154, 2017.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16:321–357, 2002.

[8] Andrea Dal Pozzolo. Adaptive machine learning for credit card fraud detection. Université libre de Bruxelles, 2015.