Transfer Naïve Bayes Learning using Augmentation and Stacking for SMS Spam Detection

Cihan Ulus, Zhiqiang Wang, Sheikh M.A. Iqbal, K.Md.Salman Khan, and Xingquan Zhu
Department of Electrical Engineering and Computer Science, Florida Atlantic University
Boca Raton, FL 33431, USA
{culus2021, zwang2022, siqbal2019, mkayamkhani2021, xzhu3}@fau.edu

***Abstract*—Short Message Service (SMS) spam, unsolicited messages delivered through phones, is common and prevalent, but difficult to filter out. Naïve Bayes (NB) classifier is a frequently used spam filtering approach for texts, due to its simple but rigorous statistical learning nature and transparency in the decision making. For SMS messages, simple NB classification is ineffective, because SMS texts are short and brief, often contain numerous typos, abbreviations, and slang words. In this paper, we propose, AstNB, a new Augmentation and Stacking combined Transfer learning approach for Naive Bayes (NB) classification. For effective transfer learning from a source domain, *e.g.* emails, to a target domain, *e.g.* SMS, AstNB first introduces data augmentation to generate different copies of training data, by combining a target domain sample with a randomly selected source domain instance, followed by training a number of basis classifiers from augmented data. After that, a stacking process is used to generate new feature space by aggregating predictions of basis classifiers and the feature space created from target data. A final classifier is trained to predict unlabeled SMS messages for spam prediction. Experiments and comparisons show that AstNB can effectively transfer knowledge from source domain for SMS spam detection, especially when the target domain has very few labeled messages.**

***Index Terms*—Transfer learning, Naive Bayes classification, short message service, SMS, spam detection.**

I. Introduction

Short Message Service, commonly known as “SMS”, is a common and effective way of communication around the world. An SMS message is comprised of short text messages within a certain length of characters, such as 160 characters [1]. Due to its short and brief nature, a short message is often noisy and contains a significant number of typos, text message abbreviations, or slang [2]. For example, “ABT” is used as an abbreviation of the word “about” (similar pronunciation), “AYS” is used to replace “are you serious” (first letter abbreviation), and “?4Y” is used to represent “question for you” (a combination of sign, pronunciation, and first letter abbreviation).

Because of its popularity in daily communication and a large number of user body, SMS platform has also drawn a large number of spammers to broadcast unsolicited messages, such as advertisement [3], spam [4], or harassment or bullying materials [5]. Spam SMS messages are unwanted SMS texts that are sent in bulk and senders usually do not have a direct relationship with receivers, and vice versa. As SMS messages are generally perceived as a personal and informal mode of communication the breach of personal information of the user is at a higher risk. Therefore, with this increasing spread of information, it has become more important than ever before to assist users in automatically determining whether incoming texts are normal or spam.

This research is sponsored by U.S. National Science Foundation (NSF) through Grants IIS-1763452 & CNS-1828181.

Several supervised learning classification techniques have been used previously to classify text as spam or normal. However, there are limitations in the existing classifiers for spam filtration of SMS messages. These limitations are because SMS messages are usually limited in their length which pose limitation in the availability of sufficient data for training classifiers and hence for testing these classifiers. Moreover, SMS spam filtering becomes difficult due to informal format of the SMS messages with the inclusion of unstandardized abbreviations, emojis, and idiosyncratic language makes it difficult for the existing classifiers to classify SMS messages [2]. Unlike emails, SMS messages do not contain headers or subject line that can help separate a legit SMS from a spam SMS [6]. Table I lists two sets of messages from two different domains, emails *vs.* SMS. The examples show that emails are not only longer in length than SMS but also have fewer abbreviations and more standardized language making them more formal and easier to understand the context and semantics.

TABLE I
Examples of “Normal” *vs.* “Spam” documents from two domains: Email and SMS

Domain	Class	Sample email/SMS content
Domain (Email)	Normal	additional recruiting i ’ m happy to introduce molly magee as the newest addition to the eops recruiting team . toni and molly have ...
	Spam	jennifer sends them to their final destination. designated as a private key 4. validate ...
Domain (SMS)	Normal	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
	Spam	Ffffffff. Alright no way I can meet up with you sooner?

Naïve Bayes (NB) classifiers are most commonly used for text classification because of their simplicity and effectiveness [7]. An NB classifier is based on the Bayes theorem, using conditional independence assumption, to classify every instance of observation into one class or another based on its probability related to the target classes [8]. Several studies [9] have proposed to use NB for detection and filtering of SMS spam messages [9]. Because of high sparsity, typos, and large vocabulary of SMS messages, it is difficult to use NB to classify them effectively. On the other hand, due to privacy and regulations, collecting a large number of SMS spam messages is often difficult, making number of training examples for SMS spam detection relatively small.

When training samples are limited, transfer learning can leverage data from other domains, and is often considered one of the most effective methods to improve the learning on a target domain [10]. Unlike traditional machine learning techniques, where it is imperative for the training and test data to have similar domain and data distributions, transfer learning addresses the problem where test and training data do not necessarily belong to the same domain but are from related domains. Many methods exist to transfer knowledge or learning models, by using feature space or model space-based approaches. For example, TrAdaBoost [11] is a model-based approach which leverages the boosting-based method to update instance weights and boosts the learning of classifiers to better classify target data. Self-taught learning [12] and cross-domain semi-supervised learning [13] repent another set of approaches which use feature-based approaches to link source and target domains for learning.

Intuitively, NB can be combined with transfer learning to classify SMS spam messages. For example, we can use NB as a TrAdaBoost base model, and train model use data from source domains (*e.g.* using emails as the source domain) to classify SMS spam. Nevertheless, as we will soon report in the experiments, this does result in good performance. NB is based on the assumption of estimation based on features distribution for example, multinomial or Bernoulli therefore NB transfer learning method alone cannot be used effectively for cases where the target dataset, like SMS dataset has a distribution different than the training dataset. For example, a novel Naïve Bayes Transfer Learning (NBTL) method has been proposed to transfer learning from the training data to test data [14]. For this purpose, Expectation-Maximization (EM) algorithm is applied to adapt the NB model, trained on training data, to the test data distribution. However, the algorithm ignores the effect of the new domain features [15]. Moreover, Li et al. propose a naive bayes based transfer learning for text classification based on group probabilities known as transfer group probability Naive Bayes (TrGNB) algorithm [16]. TrGNB is based on the group probability information of the source and target domains integrated into the Naive Bayes classification model using transfer learning. For better transfer learning, the algorithm also uses KL-divergence to measure the difference in the distribution of source and target domains [16], [17]. Unlike the existing approaches, our proposed augmentation and stacking approach is based on the augmentation of target data with source data for better transfer learning.

In this paper, we propose a new transfer learning method AstNB, which combines data augmentation and model stacking for SMS spam classification. AstNB leverages the strength of model-based and feature-based transfer learning approaches, and is particularly designed for SMS data with very limited training samples. More specifically, AstNB augments a small portion of randomly sampled source data with the target dataset to address the issues of limited length of the SMS texts. These augmented datasets are then used to train basis classifiers, whose outputs are stacked together to form a new feature space, in combination with the original feature space of the target data. A final predictive model is then trained to classify the target dataset for Spam filtration.

In summary, our research has two major contributions to advance machine learning based approaches for SMS spam detection:

- Cross-domain SMS Data Augmentation: We propose to use data augmentation based approaches to alleviate data sparsity, typos, and abbreviations, which are common challenges for SMS data. By augmenting data from other domains, such as emails, each SMS text is expanded to include more information for learning models to include words/symbols/notations that do not appeared in the SMS training data, but may appear in the test SMS data.
- Model Stacking for Transfer Learning: After obtain different sets of augmented target datasets, we train multiple base models, and stack their outputs, in combination with the feature space of the target dataset, to form a new feature space for learning. This provides a way to leverage features and models for effective transfer learning of SMS data.

II. Related work

A. Machine Learning for Spam Detection

Machine learning has been used in numerous fields for Spam detection, such as email spams and SMS spams [1]. Among all approaches, content-based methods use keywords or tokens (such as emojis) as features to train classifiers for detection. In additional to common learning methods, such as support vector machines (SVM) or multi-layer perceptrons [18], NB has also been used for detection and filtering of SMS spam messages [9]. The model was a Naive Bayes SVM (Support Vector Machines) that utilised contained process steps to clean the data extensively before separating and tokenizing it and later trained with maximum entropy where its performance was compared using class conditional independence. They conclude that SVM classifier model showed higher accuracy compared to that of baseline Naive Bayes at spam detection when utilizing their methods.

Email and SMS spam messages are becoming difficult to detect nowadays, because they are evolving to better disguise themselves as normal messages. Multiple techniques have been employed by many researchers and scientists to solve this problem with Naïve Bayes being one of the most commonly used methods on many different types of datasets.

One particular research utilize where they designed and tested Naive Bayes algorithm for Email spam filtering on two different datasets [19] *i.e.*, Spam Data and SPAMBASE datasets and test its performance. In this comparative study, the open source WEKA tool was used to perform the model design and testing. SPAMBASE was taken from UCI machine learning repository. This dataset contains 4,601 email messages and 58 attributes and is a collection of non-spam email obtained from filled work, personal Email and single email accounts. Each instance in this SPAMBASE consists of 58 attributes where most of the said attributes represent the frequency of a given word or character in the email that corresponds to the instance. These specifications make this dataset suitable for algorithm testing. The authors concluded that the Naïve Bayes classifier gave higher performance in spam detection when using SPAMBASE dataset compared to normal spam data. They stated that the dataset needs to have proper cleaning and refinement to be able to filter spam messages much better.

Several other researchers have also used different types of machine learning methods, including SVM, multinomial NB, k-nearest, random forest, and decision trees [20], [21] for SMS spam detection. The results show that NB consistently outperforms other alternatives. On the other hand, by tuning term-frequency, it is possible for random forest to outperform NB, with a very thin margin [21].

B. *Transfer learning for Spam Detection*

Transfer learning is another state-of-the-art approach commonly used by researchers in machine learning for classification of data, especially when the target data is very limited and there are some auxiliary data from other domains to support the learning. Traditionally, transfer learning is mainly used for tabular-data. For example, TrAdaBoost [11] works on two, source and target, datasets with shared feature space (*i.e.* the two datasets have the same feature space).

In addition to classical transfer learning, methods also exist to leverage neural language model based deep neural networks for spam detection. In this case, transfer learning is achieved by training a language model from one domain, then transferring the model to another domain. One study, in particular, has utilized Transfer Learning of BERT Model for universal spam detection for email messages [22] where a Universal Spam Detection Model (USDM) was built and trained with four datasets and leveraged hyperparameters from each model. The combined model was then fine-tuned with the same hyperparameters from these said four models separately.

Text classification is also an important section of spam detection that is required for proper processing to occur. One study utilized a Fine-Tuned BERT-Based Transfer Learning Approach for Text Classification purposes [23] where NLP (Natural Language Processing) was utilized for better categorizing documents containing different types of texts. Different BERT models were designed all having encoding and feature extraction with each model attaining preferable accuracy scores. The performance of these models led to the researchers concluding that NLP and text classification were essential for proper text classification using transfer learning and Naive Bayes classifiers.

III. Problem Definition and Preliminary

A. *Problem Definition*

Let T denotes a target dataset containing a number of SMS messages. We use T^l to denote the labeled subset and T^u denote the unlabeled subset of T, respectively. The size (number of samples) of a dataset is denoted by $|\cdot|$. We use $t_i \in T$ to denote a single SMS message (or document), and $y_i^t \in \mathcal{Y}^t$ represents the label of t_i. For spam detection purposes, the label of an SMS message has two classes, *i.e.* $\mathcal{Y}^t \in \{\text{Spam}, \text{Normal}\}$. In a transfer learning setting, a source domain dataset S is provided to support the learning, where $s_i \in S$ denotes a single document in S, and $y_i^s \in \mathcal{Y}^s$ denotes the label of s_i. We assume that documents in the source dataset are labeled, and their label space is binary and relevant to the target domain: $\mathcal{Y}^s \in \{\text{Spam}, \text{Normal}\}$.

Given a small number of labeled training set in the target domain, *i.e.* $|T^l| << |T^u|$, the objective of NB based transfer learning is to train a classifier with maximum performance (*e.g.*, accuracy and AUC values) in predicting unlabeled samples in the target domain T^u.

B. *NB Classification*

In this paper, we use NB as the learning algorithm. Our experiments in Section V will soon demonstrate that for SMS spam detection simple NB performs much better than other classifiers, especially when the training data are very limited.

NB is a Bayes theorem based probabilistic learning approach, which evaluates the conditional probability that a given observation belongs to a particular class. Bayesian networks (BN) are important building blocks in Naïve Bayes classifiers as they give a clear cut idea on representing knowledge about a particular domain where each node represents a variable and each edge represents the conditional probability for the corresponding variable. These probabilistic graphical models are in terms of directed acyclic graph (DAG), *i.e.*, they do not contain and self-connection or loop, as shown in Figure 1.

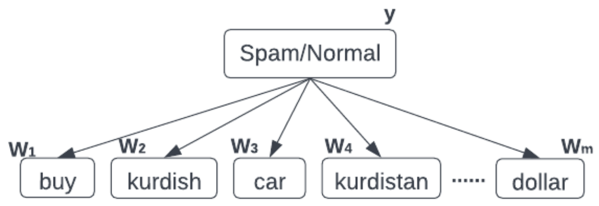

Fig. 1. Naive Bayes graphical model for SMS classification. y denotes class label, and $w_1, \cdots, w_m$ are keywords/tokens, which are conditionally independent, given class label y.

Given an SMS instance t_i, which is represented using some keywords/tokens $t_i = [w_1, w_2, \cdots, w_m]$, NB classification is to classify t_i into one of the target classes $y_i \in \{\text{Spam}, \text{Normal}\}$. Using maxim a posterior (MAP) decision theory, this is formatted as the following process:

$$y_i = \text{argmax}\, P(y_i \in \{\text{Spam}, \text{Normal}\}|t_i) \quad (1)$$

Because t_i is represented using keywords/tokens: $t_i = [w_1, w_2, \cdots, w_m]$, we have

$$y_i = \operatorname{argmax} P(y_i \in \{\text{Spam}, \text{Normal}\} | w_1, w_2, \cdots, w_m) \quad (2)$$

Using Bayes theory, we have

$$y_i = \underset{y_i \in \{\text{Spam,Normal}\}}{\operatorname{argmax}} \frac{P(w_1, \cdots, w_m | y_i) \times P(y_i)}{P(w_1, \cdots, w_m)} \quad (3)$$

Using Naive Bayes assumption, features w_i are conditionally independent given the class label y_i, the joint conditional probability $P(w_1, \cdots, w_m|y_i)$ is calculated using the product of the conditional probabilities of all features, we have

$$y_i = \underset{y_i \in \{\text{Spam,Normal}\}}{\operatorname{argmax}} \frac{\prod_{j=1}^{m} P(w_j | y_i) \times P(y_i)}{P(w_1, \cdots, w_m)} \quad (4)$$

IV. Proposed Method

SMS spam detection is an essential text classification task. By using simple notations, we can first extract a vocabulary Σ from a collection of documents, and represent each SMS document t_i as a set of keywords (or tokens), *i.e.* $t_i = [w_1, w_2, \cdots, w_m]$, where w_j denotes a keyword/token and $w_j \in \Sigma$. In a simple bag-of-word notation, each unique word in the source and target data set represents a feature of the NB classification model. For this purpose, Σ_s and Σ_t denote vocabularies of the source domain and target domain, respectively. Because source domains and target domains are related to each other, it is convincing to assume that $\Sigma_s \cap \Sigma_t \neq \emptyset$, meaning that two domains share some keywords or tokens. The niche of transfer learning stems from the fact that the training sample of the target domain is very limited, resulting in a small number of keywords from the training set. By using source data (and its vocabulary Σ_s) to expand the vocabulary space of the target set, it will help make accurate prediction of the test samples in the target set.

A. Data Augmentation

The length of the training document affects machine learning model performance in text classification. In general, longer documents contain more content and context, and will result in better accuracy. In reality, it is not always possible to have long documents, especially when dealing with short and brief documents such as SMS texts. In this paper, we use data augmentation to deal with that problem. Data augmentation is a process where we generate different copies of training data, by combining target domain data with randomly sampled source domain data. Since SMS messages are short and often contain numerous typos, abbreviations, and slang words, data augmentation can enrich SMS messages by using source data.

One of the important things to consider for augmentation is the similarity between texts that will be combined. If the target data T consists of daily talk/conversations, the source data S should come from similar domains. This will help transfer patterns from source data S and use them to classify target data T.

Denote s_i a document from source domain and t_j a document from target domain. For the ease of demonstration, assume that s_i and t_j consist of a single sentence as follows:

t_j = "how are you?"
s_i= "this time of year is extremely hectic".

Let's assume both y_j^t and y_i^s are Normal and we want to augment target document $t_1 = j$ with the source document s_i. The augmented document $\tilde{t_j}$ will be as follows:

$\tilde{t_j}$ = "how are you? this time of year is extremely hectic"

1) Target Dataset Augmentation: Using the above single instance augmentation, we can now generate a random subset of augmented instances for the target set. First of all, for each $t_i \in T^l$, we check the label of the target document $y_i^t \in \mathcal{Y}^t$. If y_i^t is a Spam, we randomly select a document from the source data s_j among the documents having y_j^s = Spam . If y_i^t is Normal, we randomly select one of the s_j among the documents having y_j^s = Normal. For each document $t_i \in T^l$ we randomly select a source document s_j, and combine the target document with the source data. By combining our input datasets t_i and s_j, we generate $\tilde{t_i}$ which is essentially the transformed/augmented data. The augmented dataset equation can be given as,

$$\tilde{t_i} = t_i \parallel random(S, y_i^t) \quad (5)$$

where $s_j = random(S, y_i^t)$ means randomly selecting one document from the source data S based on the label of t_i for the augmentation process, and $\parallel$ denotes concatenation of two documents. To avoid selecting the same source document s_j to combine for each target document, a large source dataset S is preferred. Otherwise we may have very similar augmented samples $\tilde{t_i}$ although, we will have a unique target document t_i for each augmentation process. Since the target dataset t_i is short, it would not make a big difference between augmented document $\tilde{t_i}$ in case of having the same source document s_j for the augmentation.

B. Stacking

Stacking is an ensemble learning technique applied to improve model predictions by combining outputs of multiple models and through another machine learning model. It is basically one meta-classifier, learning the output of the combined base classifiers. In other word, the stacking process generates a new feature space by aggregating predictions of basis classifiers and the feature space created from target data.

Our model AstNB utilises a combination of Stacking, Augmentation and Transfer Learning for Naïve Bayes classification. Having multiple datasets and multiple steps in the process leads to bias which is undesirable in classification. This step is done at the end since Data augmentation cannot be done on the outputs of the basis models and hence must be done early in the process. The name itself states its where we stack models one on top of another.

Stacking starts after the augmentation process is completed. We trained NB classifiers $\hbar_X(\cdot)$ with augmented documents

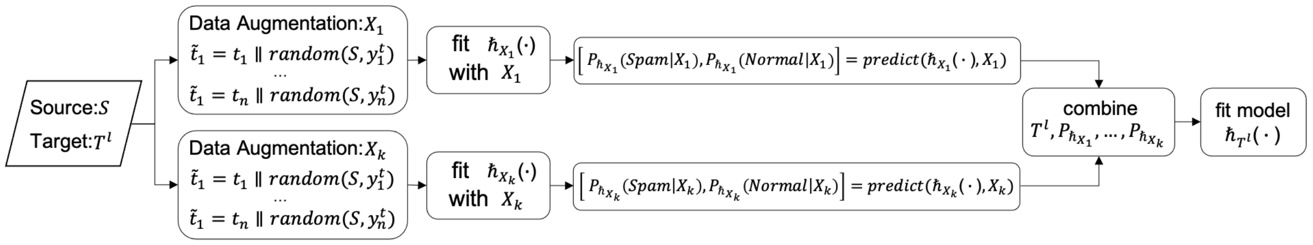
Source:S
Target:T^l
Data Augmentation:X_1
$\tilde{t}_1 = t_1 \parallel random(S, y_1^t)$
...
$\tilde{t}_1 = t_n \parallel random(S, y_n^t)$
Data Augmentation:X_k
$\tilde{t}_1 = t_1 \parallel random(S, y_1^t)$
...
$\tilde{t}_1 = t_n \parallel random(S, y_n^t)$
fit $\hbar_{X_1}(\cdot)$ with X_1
fit $\hbar_{X_k}(\cdot)$ with X_k
$\left[P_{\hbar_{X_1}}(Spam|X_1), P_{\hbar_{X_1}}(Normal|X_1)\right] = predict(\hbar_{X_1}(\cdot), X_1)$
$\left[P_{\hbar_{X_k}}(Spam|X_k), P_{\hbar_{X_k}}(Normal|X_k)\right] = predict(\hbar_{X_k}(\cdot), X_k)$
combine $T^l, P_{\hbar_{X_1}}, \ldots, P_{\hbar_{X_k}}$
fit model $\hbar_{T^l}(\cdot)$

Fig. 2. The overall framework of AstNB. From left to right: Source data S are used to generate augmented dataset: $X_1, \cdots, X_K$, each of which helps train one base model $\hbar_{X_k}(\cdot)$. The outputs of base models help train final model: $\hbar_{T^l}(\cdot)$.

in training phase. We partition the source data by labels: $\{\mathcal{S}^{Normal}, \mathcal{S}^{Spam}\}$. After that, we create a keyword feature $\mathcal{F} \in \mathbb{R}^{|T^l| \times m}$ from T^l to later use for training the final model. As a result we obtain a two dimensional array $|T^l| \times m$, where m denotes the number of the words after stopwords removal and $|T^l|$ denotes the number of labelled documents. We also create a zero array $\hat{\mathcal{F}} \in \mathbb{R}^{|T^l| \times K}$ to store the posterior probability of predicted labels, the output from the NB classifiers $\hbar_X(\cdot)$, where K denotes the number of NB classifiers $\hbar_X(\cdot)$. After applying the augmentation process, we fit submodels with X_k and create the NB classifier $\hbar_{X_k}(\cdot)$, X_k denotes the augmented documents. We use these K NB classifiers $\hbar_{X_k}(\cdot)$ to predict the posterior probability of $\hbar_X(\cdot)$ classifying instance x as Spam; $P_{\hbar_X}(\text{Spam}|x)$.Later we combine the original keywords with the posterior probability values $P_{\hbar_X}(\text{Spam}|x)$ and we obtain $\hat{\mathcal{F}} \leftarrow \mathcal{F} \parallel \hat{\mathcal{F}}$. As for the final step, we train a stacking NB classifier $\hbar_{T^l}(\cdot)$ using features $\hat{\mathcal{F}}$.

Our motivation is to utilize base model variations to minimize the bias in the output predictions. By observing the base NB classifier, notable proportions of bias are discovered. The employment of stacked ensemble model aims to tackle this issue and our experiments show that stacking is indeed efficient for minimizing bias in outputs.

1) Features for Stacking: For the features of submodels, we use words of augmented target documents $\tilde{t}_i$. For the main MNB model, the predicted probability values from each submodel, plus words from original target dataset, are used. As a result, the total number of features for stacking is the number of unique words in original dataset plus $2 * K$, where K denotes number of the submodel and each model has 2 predicted probability values as output.

C. Overall Framework

Figure 2 shows the overall framework of the proposed AstNB model. For effective transfer learning from a source domain, *e.g.* emails, to a target domain, *e.g.* SMS, AstNB first introduces data augmentation to generate different copies of training data, by combining target domain data with a small portion of randomly sampled source domain data, followed by training a number of basis classifiers from augmented data. After that, a stacking process is used to generate a new feature space by aggregating predictions of basis classifiers and the feature space created from target data. A final classifier is trained to predict test SMS message for spam prediction.

D. AstNB Algorithm

Algorithm 1 lists the detailed procedure of the proposed AstNB method. AstNB has two phases, training phase and prediction phase. During the training phase, it splits the source dataset S into two parts, S^{Normal} and S^{Spam} by labels. With the given K, AstNB algorithm creates K MNB models and for each of the MNB model, it is fitted with augmented data. During the augmentation process, each of the target documents, SMS, is combined with a random source document, email, which has the same label as the SMS. Once the K MNB models have been trained, they are used to predict the probabilities of labels for both augmented $\tilde{T^l}$ and T^u . After that, the K results of $\tilde{T^l}$ are combined with T^l as training dataset to fit the final MNB model.

During the prediction phase, AstNB uses the K trained MNB models to predict labels probabilities of T^u and then adds these results back to T^u as a augmented testing dataset, $\tilde{T^u}$. Later, the final MNB model uses $\tilde{T^u}$ to predict labels.

V. Experiments

A. Benchmark Datasets

In this paper, we use emails as the source domain and short messages (SMS) as the target domain. Table II shows the basic statistics of these two datasets. The SMS dataset (target data) is imbalanced with 4,828 (87%) normal and 747 (13%) spam texts. Compared to the SMS dataset, the Email dataset (source data) is considerably less imbalanced with 3,672 (71%) normal and 1499 (29%) spam emails. For both datasets, the character and minimum length are the same: 5 characters. However, as expected, there is a big difference in maximum and average length of both datasets. The maximum and average length of emails are 31,186 and 1,024 characters, respectively, whereas SMS data has a maximum length of 913 characters and an average length of 83 characters.

B. Experimental Settings

We utilize stratified k-fold cross validation techniques but inverse the training *vs.* test data. For example, 0.2% of the training data is achieved using 500-fold cross validation with 1-fold being used as training and rest data being used as test

Algorithm 1: AstNB: Stacking and Augmentation Transfer Naive Bayes Learning

```
Data: (1) $\mathcal{S}$: source dataset (Email);
      (2) $T^l$: Labeled target dataset (SMS);
      (3) $T^u$: Target dataset for prediction (SMS)
Define: (1) $\hbar_X(\cdot)$: An NB classifier trained from dataset $X$;
        (2) $\hbar_X(x)$: Predicted label of $\hbar_X(\cdot)$ on instance $X$;
        (3) $P_{\hbar_X}(\text{Spam}|x)$: Probability $\hbar_X(\cdot)$ classifying
            instance $x$ as Spam.
Input: $K$: Stacking size.
Output: $[y^t_1, \cdots, y^t_{|T^u|}]$: Predicted labels of $T^u$.

=== Training phase ===
$\{\mathcal{S}^{Normal}, \mathcal{S}^{Spam}\} \leftarrow$ Partition source dataset $\mathcal{S}$ by labels.
$\mathcal{F} \in \mathbb{R}^{|T^l| \times m} \leftarrow$ Create keyword features from $T^l$;
$\hat{\mathcal{F}} \in \mathbb{R}^{|T^l| \times 2K} \leftarrow \mathbf{0}$;
for $k = 1, \cdots, K$ do
    $X_k \leftarrow \{\}$
    for each target instance $t_i = 1, \cdots, |T^l|$ do
        if $y^t_i$ == Spam then
            $s_j \leftarrow$ random sample from $\mathcal{S}^{Spam}$
        end
        else
            $s_j \leftarrow$ random sample from $\mathcal{S}^{Normal}$
        end
        $\tilde{t}_i \leftarrow t_i \parallel s_j$;
        $X_k \leftarrow X_k \cup \tilde{t}_i$;
    end
    $\hbar_{X_k}(\cdot) \leftarrow$ Train an NB classifier from $X_k$;
    for each instance $\tilde{t}_i = 1, \cdots, |X_k|$ do
        $\hat{\mathcal{F}}[i, 2k] \leftarrow P_{\hbar_{X_k}}(\text{Spam}|\tilde{t}_i)$;
        $\hat{\mathcal{F}}[i, 2k+1] \leftarrow P_{\hbar_{X_k}}(\text{Normal}|\tilde{t}_i)$
    end
end
$\hat{\mathcal{F}} \leftarrow \mathcal{F} \parallel \hat{\mathcal{F}}$;
$\hbar_{T^l}(\cdot) \leftarrow$ Train a stacking NB classifier using features $\hat{\mathcal{F}}$;

=== Prediction Phase ===
for each target instance $t_j = 1, \cdots, |T^u|$ do
    $\mathcal{F}_j \in \mathbb{R}^{1 \times m} \leftarrow t_j$'s keyword features from $\mathcal{F}$;
    $\hat{\mathcal{F}}_i \in \mathbb{R}^{1 \times 2K} \leftarrow [P_{\hbar_{X_1}}(\text{Spam}|t_j), \cdots, P_{\hbar_{X_K}}(\text{Spam}|t_j)]$;
    $\hat{\mathcal{F}}_j \leftarrow \mathcal{F}_j \parallel \hat{\mathcal{F}}_j$;
    $y^t_j \leftarrow \hbar_{T^l}(t_j)$;
end
Return $[y^t_1, \cdots, y^t_{|T^u|}]$
```

TABLE II
BASIC STATISTICS OF TWO DATASETS

Dataset	# of Normal	# of Spam	Max Length	Min Length	Ave.
Email	3,672	1,499	31,896	5	1,024
SMS	4,828	747	913	5	83

data. This allows us to test algorithm performance using very few training data.

C. Baselines

We compare the performance of the proposed method AstNB with the following three types of baseline methods:

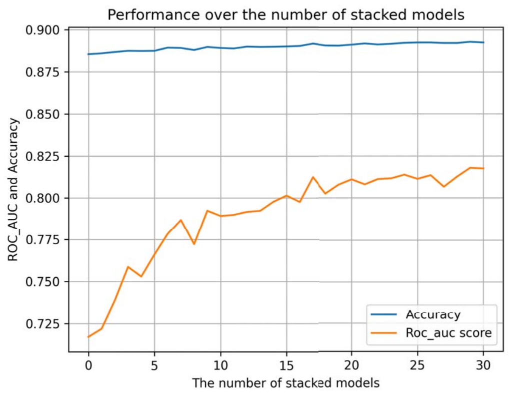

Fig. 3. Performance over the number of stacked models using 0.2% training dataset for training.

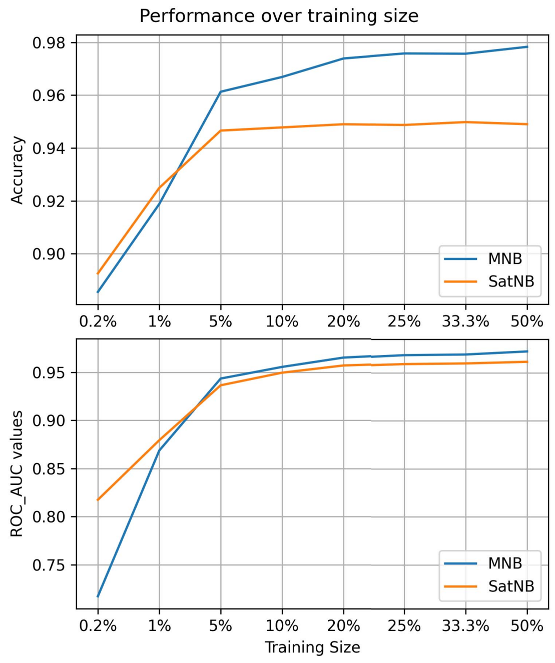

Fig. 4. Performance over training size for MNB and AstNB which has 30 stacked models

Single Classifier Baseline:

- *MultinomialNB (MNB):* Multinomial Naive Bayes models from sklearn class package for classification with discrete features *e.g.*, word counts for text classification.
- *RidgeClassifier (RC)*: This is a frequently used baseline which considers text classification as a regression task.
- *DecisionTree (DT)*: Decision trees are frequently used for text classification due to their good transparency and interpretations.
- *Support vector machines (SVM)*: SVM is a strong machine learning method for classification, especially for numeric data.

Ensemble Classifier Baseline:

- *BaggingClassifier*: Bagging fits base classifiers from random subsets of the original dataset and then aggregates their individual predictions for final prediction.
- *RandomForestClassifier (RF)*: RF learns decision trees from sub-samples and sub-features of the original dataset, and uses combined results from trees for final prediction.

Transfer Learning Classifier Baseline:

- *TrAdaBoost [11]:* A classical transfer learning method extends boosting-based learning algorithms to utilize a small amount of newly labeled data to leverage the old data to construct a high quality classification model for the new data. TrAdaBoost is also an ensemble method which can be used in combination of different types of base classifiers.
- *Transfer Naive Bayes (TNB) [24], [25]:* A modification of Naive Bayes using gravitational weighting.
  - **TNB+**S: This method uses S as source data, and T^l as target data to fit the model, and predict T^u. This is a classical transfer learning setting for SMS prediction.
  - **TNB**$\neg S$: This model uses T^l as source and target domain data, without using S, to fit the model and predict T^u.

    We introduce two variants of TNB, including TNB+S and TNB$\neg S$, to compare a transfer learning framework with *vs*. without leveraging data from source domains.

D. Performance Comparison

1) Transfer Learning Results: Table III reports the performance of different methods in a tyical transfer learning setting (using 0.2%, 1%, and 5% target data as training data). The AstNB is based on 30 stacked models, and TrAdaboost and BaggingClassifier ran with 30 estimators. All ensemble methods use the same number of base models to avoid taking advantage of utilizing more base models. In a transfer learning setting, target domain often has very few samples, and the nature of SMS also makes it difficult to collect a comprehensive training set (because users vary dramatically in their writings). Therefore, we use a small percentage of training set sizes.

As we can see from Table III, AstNB has much better results than other methods. In terms of accuracy, although AstNB slightly outperforms MNB and RidgeClassifier, it outperforms all of other models. However, accuracy is not a good metric for the performance of the models since the data we have is very unbalanced. If the model predict all the short messages as normal, it will have 0.87 accuracy since 87% of all short messages are normal. That's why the main criteria for the performance comparison is AUC value of the models. In terms of AUC, AstNB outperforms all other methods when we use 0.2% and 1% data as training data. When we have 5% training data MNB performs better than AstNB. 5% can be seen as a threshold to have enough original data to develop a model without augmentation and perform better than AstNB which uses augmented data.

In our experiments, TNB is significantly inferior to other methods, especially when including source domain data S. Transfer learning using source data S inverse impacts on its performance. Without including S, TNB$\neg S$ actually achieve much better performance, but is still inferior to AusNB. In addition, TNB is also computationally expensive. When training TNB using emails, its runtime cost is over the sum

TABLE III
ALGORITHM PERFORMANCE COMPARISON. 0.2%, 1%, AND 5% DENOTE TRAINING SET SIZE COMPARING TO WHOLE DATASET.

Model	Measures	0.2%	1%	5%
MNB	Accuracy	0.8855	0.9188	0.9613
	AUC_ROC	0.7172	0.8687	0.9435
AstNB	Accuracy	**0.8925**	**0.9249**	0.9466
	AUC_ROC	**0.8175**	**0.8794**	0.9365
TNB+S	Accuracy	0.2242	0.3130	0.5588
	AUC_ROC	0.4703	0.3935	0.1874
TNB$\neg S$	Accuracy	0.8662	0.8703	0.9452
	AUC_ROC	0.5470	0.8009	0.9499
Tradaboost (30 estimators)				
MNB	Accuracy	0.5973	0.7459	0.9114
	AUC_ROC	0.7323	0.8024	0.8762
RidgeClassifier	Accuracy	0.8676	0.8744	0.9238
	AUC_ROC	0.5058	0.5324	0.7181
DecisionTree	Accuracy	0.2968	0.4291	0.7521
	AUC_ROC	0.5484	0.5971	0.7260
SVM	Accuracy	0.4280	0.8815	0.8762
	AUC_ROC	0.5120	0.5621	0.7380
RandomForest	Accuracy	0.2931	0.5225	0.8534
	AUC_ROC	0.5541	0.6441	0.7838
BaggingClassifier (30 estimators)				
MNB	Accuracy	0.8798	0.9197	0.9634
	AUC_ROC	0.7125	0.8850	0.9519
RidgeClassifier	Accuracy	0.8671	0.8703	0.9090
	AUC_ROC	0.5043	0.5164	0.6608
DecisionTree	Accuracy	0.8669	0.8758	0.9120
	AUC_ROC	0.6375	0.7794	0.9087
SVM	Accuracy	N/A	0.8680	0.8907
	AUC_ROC	N/A	0.7569	0.8825
RandomForest	Accuracy	0.8660	0.8662	0.8736
	AUC_ROC	0.7025	0.8600	0.9540

TABLE IV
PERFORMANCE COMPARISON - SINGLE CLASSIFIER

Model	Mea.	0.2%	1%	5%	10%	80%	90%
MNB	Acc	**0.881**	**0.923**	**0.961**	**0.968**	**0.982**	**0.981**
	AUC	0.704	0.876	0.945	0.956	0.979	0.979
DT	Acc	0.869	0.886	0.916	0.926	0.964	0.9656
	AUC	0.524	0.607	0.733	0.799	0.906	0.907
RF	Acc	0.866	0.868	0.895	0.922	0.969	0.972
	AUC	0.659	0.852	0.948	0.963	0.987	0.986
SVM (p=True)	Acc	0.866	0.867	0.879	0.902	0.974	0.976
	AUC	0.449	0.786	0.899	0.931	0.979	0.980
SVM (p=False)	Acc	0.866	0.867	0.8790	0.902	0.974	0.976
	AUC	0.500	0.505	0.548	0.635	0.907	0.912
RC	Acc	0.867	0.874	0.923	0.943	0.977	0.978
	AUC	0.505	0.532	0.716	0.789	0.915	0.919

of all other seven models.

2) AstNB Performance w.r.t Base Models: The Fig.3 illustrates that the value of ROC_AUC score (the orange curve) increases dramatically over 30 stacked models from 0 to 30 while the value of accuracy (the blue curve) only has a slightly increase.

When there is no stacked model, AstNB becomes a MNB model. From Fig. 3, the ROC_AUC score starts from 0.7172 of the MNB model, and the value rises significantly to 0.7801 when the number of stacked models is 6. After that, the curve starts to saturate to 0.8271 when AstNB has 30 stacked models. This ROC_AUC score is improved significantly, by approximately 15.32%. The 0.8271 ROC AUC score is desirable being closer to 1.

3) AstNB Performance w.r.t. Training Set Sizes:: Fig. 4 shows that AstNB (orange line) has better performance than MNB (blue line) both in accuracy and ROC_AUC value at the beginning while MNB outperforms AstNB soon after 1% and remain the dominant position for the rest of training sizes.

As for ROC_AUC score, AstNB starts from 0.8175 which is about 14% higher than MNB, 0.7172, when training size is 0.2%. AstNB continues having better performance than MNB with 1% training data, while MNB outperforms AstNB when training size starts 5%. After that, MNB and AstNB have the same trend that both grow up very slowly before they reach the high to 0.9723 and 0.9615, respectively.

With regards to the accuracy of AstNB, it begins at 89.25% and increases dramatically to 92.49% with 1% training data before reaching to a high of 94.66%. After that, it stays at around 95% for the rest of training size. Likewise, MNB has a relative low accuracy of 88.55% at 1% training data size then it rises up sharply to 96.13% at 5% size which already surpasses AstNB. After that, the difference between MNB and AstNB continues to increase slowly to 1.08% before MNB peaks to 97.23%.

Table IV shows that MNB has better performance than all other baseline classifiers. This concludes that NB is indeed one of the top choices for SMS spam detection.

VI. Conclusion

In this paper, we proposed a new stacking and augmentation combined NB transfer learning approach for short message (SMS) spam detection. We argued that SMS texts are sparse, and contain many typos, abbreviations, and slang words, making spam detection difficult especially when the number of training samples is very limited. To tackle the challenges, we introduced data augmentation to enrich the training SMS data, by using data from other domains, followed by stacking predictions from NB models trained from augmented data to train another model for final prediction. We validated the performance of the proposed method by using emails as the source dataset. The results showed the effectiveness of the proposed designs.

In our future work, we are seeking to investigate neural language models and deep neural network based transfer learning framework to improve SMS spam detection.

References

[1] S. J. Delany, M. Buckley, and D. Greene, "Sms spam filtering: Methods and data," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9899–9908, 2012.

[2] O. Abayomi-Alli, S. Misra, and A. Abayomi-Alli, "A deep learning method for automatic sms spam classification: Performance of learning algorithms on indigenous dataset," *Concurrency and Computation: Practice and Experience*, p. e6989, 2022.

[3] X. Zhu, H. Tao, Z. Wu, J. Cao, K. Kalish, and J. Kayne, *Fraud Prevention in Online Digital Advertising*. Springer, 2017.

[4] H. A. Najada and X. Zhu, "isrd: Spam review detection with imbalanced data distributions," in *Proceedings of the 15th IEEE Intl. Conf. on Information Reuse and Integration (IRI 2014)*, 2014, pp. 553–560.

[5] S. Wang, X. Zhu, W. Ding, and A. A. Yengejeh, "Cyberbullying and cyberviolence detection: A triangular user-activity-content view," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, pp. 1384–1405, 2022.

[6] C. B. Asaju, E. J. Nkorabon, and R. O. Orah, "Short message service (sms) spam detection and classification using naïve bayes." in *International Journal of Mechatronics, Electrical and Computer Technology (IJMEC)*, vol. 11, 2021, pp. 4931–4936.

[7] H. S. P. J. Abdi A., Shamsuddin SM, "Deep learning-based sentiment classification of evaluative text based on multi-feature fusion," *Information Processing & Management*, vol. 56, no. 4, pp. 1245–1259, 2019.

[8] S.-J. Bang and W. Wu, "Naïve bayes ensemble: A new approach to classifying unlabeled multi-class asthma subjects," in *IEEE Intl. Conf. on Bioinformatics & Biomedicine*, 2016, pp. 460–465.

[9] P. Navaney, G. Dubey, and A. Rana, "Sms spam filtering using supervised machine learning algorithms," in *8th Intel. Conf. on Cloud Comput., Data Science & Eng.* IEEE, 2018, pp. 43–48.

[10] P. S. Jialin and Y. Qiang, "A survey on transfer learning," *IEEE Trans. on Knowledge and Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[11] Y. Q. Dai Wenyuan, X. Guirong, and Y. Yong, "Boosting for transfer learning," in *Proc. of the 24th Intl. Conf. on Machine Learning, Corvallis, USA*, 2007, pp. 193–200.

[12] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proc. of the 24th Intl. Conf. on Machine Learning*, 2007, p. 759–766.

[13] X. Zhu, "Cross-domain semi-supervised learning using feature formulation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 6, pp. 1627–1638, 2011.

[14] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Transferring naive bayes classifiers for text classification," in *AAAI*, 2007, pp. 540–545.

[15] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer learning using computational intelligence: A survey," *Knowledge-Based Systems*, vol. 80, pp. 14–23, 2015.

[16] J. Li, W. Wu, and D. Xue, "Transfer naive bayes algorithm with group probabilities," *Applied Intelligence*, vol. 50, no. 1, pp. 61–73, 2020.

[17] L. Fei and Y. Deng, "A new divergence measure for basic probability assignment and its applications in extremely uncertain environments," *Intl. Journal of Intelligent Systems*, vol. 34, no. 4, pp. 584–600, 2019.

[18] H. Kaur and P. Verma, "E-mail spam detection using refined mlp with feature selection," vol. 9, 09 2017, pp. 42–52.

[19] N. Rusland, N. Wahid, S. Kasim, and H. Hafit, "Analysis of naïve bayes algorithm for email spam filtering across multiple datasets," *IOP Conference Series: Materials Science and Engineering*, vol. 226, p. 012091, 08 2017.

[20] H. Shirani-Mehr, "Sms spam detection using machine learning approach," in *Stanford University, CS229: Machine Learning*, 2013.

[21] N. N. Amir Sjarif, N. F. Mohd Azmi, S. Chuprat, H. M. Sarkan, Y. Yahya, and S. M. Sam, "Sms spam message detection using term frequency-inverse document frequency and random forest algorithm," *Procedia Computer Science*, vol. 161, pp. 509–515, 2019.

[22] V. S. Tida and S. Hsu, "Universal spam detection using transfer learning of bert model," 2022. [Online]. Available: https://arxiv.org/abs/2202.03480

[23] R. Qasim, W. H. Bangyal, M. A. Alqarni, and A. A. Almazroi, "A fine-tuned bert-based transfer learning approach for text classification," *Journal of Healthcare Engineering*, 2021.

[24] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," *Information and Software Technology*, vol. 54, no. 3, pp. 248–256, 2012.

[25] Ashtonwebster and Z. Wang, "Transfer learning algorithm library," https://github.com/ashtonwebster/tl_algs, 2022.