Report for AI Project: SPAM SMS Filter

Zhichuang Sun (SBUID:110345185)
Shihe Lyu(SBUID:111145582)

Introduction

Problem Definition

Short Message Service is a text communication service come along with the popularity of mobile phones. Due to its low cost, it becomes a big target of the advertisers. Almost everybody who use mobile phones has received spam SMS messages, just like spam e-mail.

The problem can be described as this, we want to build a system, for which the input is a SMS message, the output is that whether it is a spam SMS or not.

Motivation

SPAM SMS is so common that everybody who use a mobile phone could have received it. It's annoying and could happen any day on .anyone. Machine Learning is a powerful tool that is suitable for this kind of text classification task. We choose this problem because

- it's a problem that people are familiar with and the result would be easy to understand and evaluate. It's a good arena for demonstrating the power of Machine Learning.
- We also want to explore more machine learning techniques and models, and learn which model is suitable for which kind of task,

so we will be more well prepared for applying the Machine Learning skills to the future challenging tasks. As a by-product, we will have a working spam SMS detector, which can be embedded into our smart-phones and do some real work! However, for this project, we will focus on the Machine Learning part.

Contributions

We are building a SPAM SMS Filter system. In this system, we focus on explore different classification models in machine learning. Generally, this system accept the SMS messages as input and preprocess the message before they are used for feature extraction, and then we apply different models with different parameters, and see which configuration is will have the best result. We will also analyze why a certain model is more suitable for this task.

The algorithms that we have used includes: Multinomial Naive Bayes, Gaussian Naive Bayes, Bernoulli Naive Bayes, K Neighbors Classifier, Decision Tree Classifier, Bagging Classifier, Support Vector Machine. We evaluate different algorithms with the following well-known

performance measures: Spam Caught(SC%), Blocked Hams(BH%), Accuracy(Acc%), and Matthews Correlation Coefficient.

What We have Learned

In order to achieve good results with machine learning, on one hand, we should dig deep into the problem and find out the most valuable preprocessing techniques and features; on the other hand, we should be familiar with the models that we are choosing, and the assumptions that the model has on the problem. For this SMS Spam Filter, we learnt the following things:

- Different models may need different preprocessing configuration. For example, one kind of tokenization may be suitable for Naive Bayes, while another kind may be suitable for Support Vector Machine.
- Train Effort can vary from model to model even for the same data. For example, Bernoulli Naive Bayes may take a lot more time to train than Multinomial Naive Bayes.
- KNeighborsClassifier is not suitable for SPAM SMS filter, and the big the K is , the worse the result.
- Support Vector Machine and Multinomial Naive Bayes did a good job in this task.

Description

Here I am going to introduce how the system is built up in detail. About implementation, there are several important things to mention here, hope this will help you understand our system or follow our implementation easily.

- We built our system with the scikit-learn, a Machine Learning Library in Python
- We followed the paper “Contributions to the Study of SMS Spam Filtering: New Collection and Results”(Almeida, DocEng11)
- Data is downloaded from the UCI’s Open Machine Learning Dataset website: https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection.

The general workflow follow like Fig-1.

Preprocessing

For text classification task, a general preprocessing includes turn all character into lowercase, but for this special task, considering SMS usually contains phone numbers, which might be an important pattern(for example, spam SMS may contain certain service number, which can differ from ads to ads, however, the bits of numbers may carry some information), in order to preserve this pattern, we can substitute any digit with character ‘N’ during preprocessing.

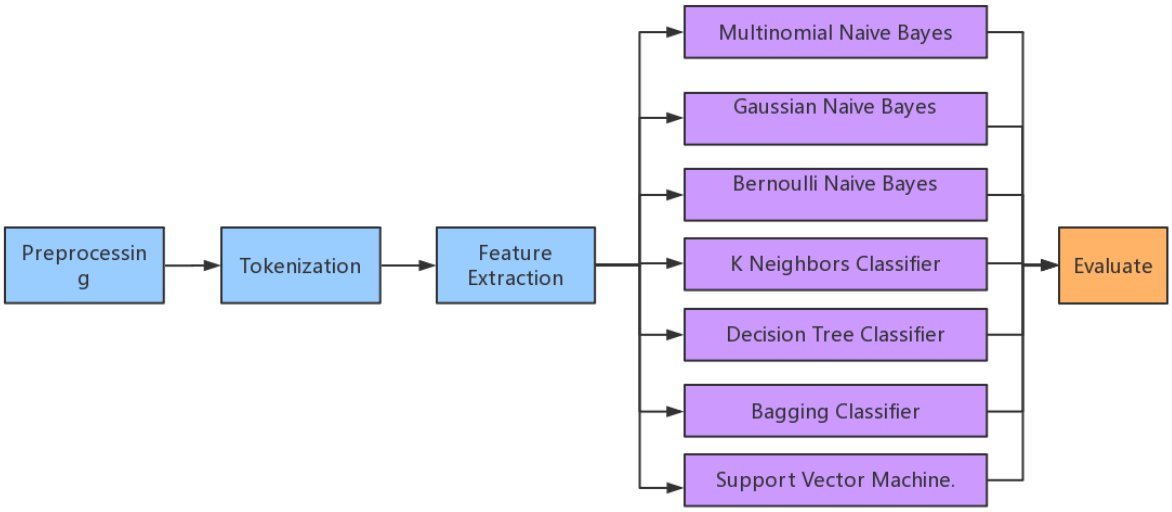

Fig1. Spam SMS Filter System Workflow

Other techniques like stop words and stemming, which are usually quite useful in text classification task, don't apply here. Because for SMS messages, they are quite short, and people often use informal abbreviations, like "Where r u? Plz call me back"

Tokenization

Consider the fact that, SMS may contain website url, mail address and other contact information, so we should be careful when we do tokenization. For example, whether we want use the whole domain names in the mail address as feature or we want to treat the subdomain as independent feature? There is no absolute answer for this question, we offer two tokenizer and will choose different tokenizer for different models:

- tok1: tokens start with a printable character, followed by any number of alphanumeric characters, including dots, commas and colons from the middle of the pattern. With this pattern, domain names and mail addresses be treated as a whole token.
- tok2: any sequence of characters separated by blanks, tabs, returns, dots, commas, colons and dashes are considered as tokens. This simple tokenizer intends to preserve other symbols that may help to separate spam and legitimate messages. [1][1]

[1] Almeida, DocEng11, Contributions to the Study of SMS Spam Filtering: New Collection and Results

Feature Extraction

We follow the text classification tradition, and build a dictionary of the words that appeared in the training data, and map each SMS message to a feature vector, using the word frequency as the feature value. This is done with the help of CountVectorizer in sklearn.feature_extraction.text library.TF-IDF is a famous feature representation, however, for SMS message which are usually quite short(less than 140 characters), it doesn't make any sense to use such a feature representation, because normally every world would appear once.

Select Models

We treat exploring different models as one of our aims, so here we are going to introduce those models we have tried, we focus on assumption behind the model and the configuration of models. We leave the evaluation to the following sections.

Multinomial Naive Bayes

Classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice).

Gaussian Naive Bayes

Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian.

Bernoulli Naive Bayes

We need to binarize the input. We can set a threshold for the binarizing.

K Neighbors Classifier

K is a set by user. We need to tune this parameter.

Decision Tree Classifier

Bagging Classifier

Bag complex classifier, like decision tree usually have good performance.

Support Vector Machine.

RBF kernel or Linear Kernel, is a configuration choice.

Evaluation

We use the dataset shared on UCI website for training and testing. There are about 5570 SMS messages collected by researchers in total. 30% of them will be used for training, the other 70% are used for testing.

For all models, we will try two different tokenizers, for certain models like K Neighbors Classifier, we will try different K values to find best configurations.

We use Spam Caught (SC%), Blocked Ham(BH%), Accuracy (Acc%), and Matthews Correlation Coefficient(MCC) as measures. The result is ranked by MCC.

Tune Model Parameters

K Neighbors Classifier ‘s performance varies with the change of K, let’s take a look at how the performance changes with number K.

Classifier	**SC%**	**BH%**	**Acc%**	**MCC**
KNeighborsClassifier+tok1,k=2	0.591	0.001	0.944	0.740
KNeighborsClassifier+tok1,k=1	0.645	0.013	0.943	0.725
KNeighborsClassifier+tok1,k=4	0.439	0.000	0.926	0.634
KNeighborsClassifier+tok1,k=3	0.369	0.002	0.915	0.567
KNeighborsClassifier+tok1,k=6	0.227	0.000	0.892	0.449
KNeighborsClassifier+tok1,k=5	0.214	0.003	0.891	0.415

As we can see, when K=2, we get the best result.
SVM can use linear kernel and rbf kernel. Let’s take a look at the difference:

Classifier	**SC%**	**BH%**	**Acc%**	**MCC**
Support Vector Machine+tok1 kernel:linear	0.889	0.003	0.983	0.924
Support Vector Machine+tok1 kernel:rbf	0.787	0.013	0.960	0.823

From the results above, we know, linear kernel is more suitable for this kind of text classification.

Results

Classifier	**SC%**	**BH%**	**Acc%**	**MCC**	
MultinomialNB+tok1	0.947	0.006	0.988	0.944	
MultinomialNB+tok2	0.910	0.005	0.984	0.929	
Support Vector Machine+tok1		0.906	0.004	0.983	0.928
Support Vector Machine+tok2		0.884	0.001	0.983	0.926
DecisionTreeClassifier+tok2	0.879	0.006	0.978	0.905	
DecisionTreeClassifier+tok1	0.880	0.009	0.977	0.896	
BernoulliNB+tok1	0.789	0.001	0.972	0.872	
BernoulliNB+tok2	0.769	0.001	0.968	0.858	
GaussianNB+tok2	0.901	0.078	0.919	0.719	
KNeighborsClassifier+tok2	0.550	0.000	0.938	0.715	
KNeighborsClassifier+tok1	0.526	0.001	0.936	0.697	

GaussianNB+tok1	0.880	0.084	0.911	0.688
Ensemble Bagging KNeighborsClassifier+tok1	0.445	0.000	0.926	0.639
Ensemble Bagging KNeighborsClassifier+tok2	0.368	0.000	0.915	0.579

tok1 = u'(?u)\\b\\w+\\w*\\b'
tok2 = u'(?u)\\b\\w+[\\-\\.\\,\\:]*\\w*\\b'

Observations

- MultinomialNB+tok1 achieved very good results, Spam Caught is high(95%), Accruacy is also high(99%), and Ham Blocked is low(0.6%)
- MultinomialNB and SVM are promising models
- BernoulliNB, KKNeighborsClassifier, Ensemble Bagging KNeighborsClassifier turn a blind eye on spam
- Tok1, which split at dot, colon, dash, and so on, is generally better than tok2, which preserve the strings concatenated by dot, colon, dash. Except for GaussianNB
- GaussianNB is too strict on spam, and mistaken a lot ham for spam

Conclusions

Spam SMS Message Filtering is a typical text classification problem, however, it is different from traditional text classification. The message is very short, the spam has its own feature, website url, phone number, email address, can all be indicators of spam message. In order to get good result, we should always start thinking from the problem's specialties, thus can we capturing useful features.

We also explored a lot of different models, some may look like each other, but they in fact have different assumption on the problem. For example, BernoulliNB and MultinomialNB are both Naive Bayes, but BernoulliNB's assumption is that input data should fit Bernoulli distributions, features are better to be binary, and the result is quite different for this task. MultinomialNB is far more better than BernoulliNB. So we should be very careful when choosing models.

Model parameters is also very important to the results. For KNeighborsClassifier, compare when K is 10 and 1, the result is quite different. For SVM, using a linear kernel or RBF kernel also matters a lot.