Contents

Introduction

True loneliness is when you don't even receive spam.

Chapter 1

Statement of the Problem

1.1 Scope and Applications

A social networking service (or SNS) is a platform to build social networks or social relations among people who share similar interests, activities, backgrounds or real-life connections (definition from Wikipedia). The ultimate purpose of any social networking service is fast and efficient exchange of information, often with intention to present it to the largest audience possible.

The vast majority of most popular social network services rely on text messages as the main form of exchange of information. Because of their openness, social networks can be extremely useful in spreading malicious messages across wide audiences, both via private (addressed to a particular individual) and public messages. Conceptually social network spam is no different from e-mail spam as private messaging services of popular social networks are equivalent in their functionality to e-mail. Hence, we can generalize the problem of classifying spam messages for both these cases.

In general, the problem of spam detection depends heavily on the application's domain and can benefit from additional metadata available along with the text message. For example, in case of e-mails the mail header is the source of metadata. Modern spam filtering systems detect the vast majority of malicious mails by simply checking the sender's reputation before proceeding with analysis of the message body.

This, of course, applies to all messaging services. Maximum effectiveness can not be achieved without using all available data in addition to the message text.

However, in most cases text analysis is the second stage preceded by a domain-specific filter. Therefore, we can further focus on statistical classification of spam for text messages without specific constraints.

Let us denote the set of all messages by M, and let $S \subseteq M$ be the set of spam messages and $L = M \setminus S$ be the set of legit messages. The ultimate goal is to obtain a decision function $f : M \to \{S, L\}$ that would determine whether a given message $m \in M$ is spam ($m \in S$) or legitimate mail ($m \in L$).

We shall look for this function by training a number of machine learning algorithms on a set of pre-classified messages $\{(m_1, c_1), (m_2, c_2), ..., (m_n, c_n)\}, m_i \in M, c_i \in \{S, L\}, 1 \leq i \leq n$. There are two aspects for the case of text messages: we have to extract features from text strings and we may have strict requirements for the precision of classifier.

1.2 Feature Selection

The entities we need to classify are text messages that are given in the form of strings. Raw strings are not convenient objects to handle in this case. Most machine learning algorithms can only classify numerical objects or otherwise require a distance metric or other measure of similarity between the objects.

Before proceeding with machine learning we have to convert all messages to numerical vectors, so called feature vectors, and then classify these vectors. The simplest example of a feature vector is the vector of the numbers of occurrences of certain words in a message.

Extraction of features usually means that some information from the original message is lost. On the other hand, the way feature vector is chosen is crucial for the performance of the filter. If the features are chosen so that there may exist a spam message and a legitimate mail with the same feature vector, then any machine learning algorithm will make mistakes no matter how good it is. A wise choice of features will make classification much easier while also fast. In most practical applications the most basic vector of word frequencies or its modification is used.

Note that at the stage of feature selection it is possible to include the features from the available metadata along with features from message text. In practice, however, it is much more important what features are chosen for classification than what classification algorithm is used.

Now let us consider those machine learning algorithms that require distance metric or scalar product to be defined on the set of messages. There does exist a suitable metric (edit distance), and there is a nice scalar product defined purely for strings [2], but the complexity of the calculation of these functions is sometimes too restrictive to use them in practice. So in this work we shall simply extract the feature vectors and use the distance/scalar product of these vectors.

Chapter 2

Basic Methods

2.1 Naive Bayes Classifier

Consider the simple case of text classification based on the presence or absence of just one word W. Suppose that we know that the word W only occurs is spam messages. This gives us confidence that any message containing W is spam. This approach can be generalized to the probability of a message feature vector occurring in the message.

Suppose we have two classes L and S corresponding to legitimate and spam messages, and that there is a known probability distribution of feature vectors $P(x|c), c \in \{L, S\}$. In general it is hard to define such distribution, but it is often possible to provide an approximation. What we need to obtain is the class that the given message belongs to, or the probability $P(c|x)$. This can be done using the Bayes' formula

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} = \frac{P(x|c)P(c)}{P(x|L)P(L) + P(x|S)P(S)}.$$

where $P(x)$ is the a-priori probability of a message with feature vector x and $P(c)$ is the probability of class c, i.e. the probability of any given message to belong to c. Given the values $P(c)$ and $P(x|c)$ for $c \in \{L, S\}$ one can calculate the probability $P(c|x)$ which cal then be used in a classification rule.

The most basic classification rule is to classify message to the category with bigger probability.

Definition 2.1.1. *Maximum a-posteriori probability (MAP) rule: if* $P(S|x) > P(L|x)$ *then classify* x *as spam, otherwise classify as legitimate message.*

The MAP rule can be transformed to

If $\frac{P(x|S)}{P(x|L)} > \frac{P(L)}{P(S)}$ then classify x as spam, otherwise as legitimate message.

The ratio $\frac{P(x|S)}{P(x|L)}$ is known as the *likelihood ratio* and is denoted as $\Lambda(x)$.

This approach can be too simplistic for certain applications. For example, in case of e-mail spam filtering, false positives (classifying legitimate message as spam) are usually much more unwanted than false negatives (classifying spam as legitimate message). The following generalization allows to take such restrictions into account.

Definition 2.1.2. *A cost function $\mathcal{L}(c_1, c_2)$ denotes the cost of misclassifying a message of class c_1 as the one belonging to class c_2.*

It is natural to put $\mathcal{L}(L, L) = \mathcal{L}(S, S) = 0$, but in general it might not be the case.

Then we can express the expected risk of classifying a message x belonging to class c in the above terms.

Definition 2.1.3. *The function $R(c|x) = \mathcal{L}(S, c)P(S|x) + \mathcal{L}(L, c)P(L|x), x \in M, c \in \{L, S\}$ is called the risk function.*

Now we can define a natural classification rule in terms of expected risk.

Definition 2.1.4. *Bayes' classification rule: if $R(S|x) < R(L|x)$ then classify x as spam, otherwise as legitimate message [3].*

It can be shown that Bayesian classifier f minimizes the average risk

$$R(f) = \int \mathcal{L}(c, f(x))dP(c, x) =$$
$$P(L) \int \mathcal{L}(L, f(x))dP(x|L) + P(S) \int \mathcal{L}(S, f(x))dP(x|S)$$

so in this sense Bayesian classifier already is optimal [1].

Naturally, the loss of classifying the message correctly is zero, thus $L(S, S) = L(L, L) = 0$. Then the Bayes's classification rule can be rewritten as

$$\text{If } \Lambda(x) > \lambda\frac{P(L)}{P(S)} \text{ classify as spam otherwise as legitimate message.}$$

Here $\lambda = \dfrac{\mathcal{L}(L,S)}{\mathcal{L}(S,L)}$ is the additional parameter that specifies the risk of misclassifying legitimate messages as spam. As the value of λ increases, the classifier produces fewer false positives.

While the bare classification process is straightforward, the practical applications of Bayes's classifier are limited by our ability to approximate the a-priori probabilities $P(x|c)$ and $P(c), c \in \{L, S\}$ from the training data. Therefore, while the Bayes's classifier is optimal in the sense of minimizing the loss of classification for given a-priori probabilities, the quality of spam detection depends on the feature selection and approximation of these probabilities.

$P(L)$ and $P(S)$ can be easily approximated by the ratio of legitimate and spam messages respectively. $P(x|c)$ is non-trivial and depends on the contents of selected feature vector. Consider the simplest case where the feature vector x_w is 1 if the message contains w and 0 otherwise. Then the probability $P(x_w = 1|S)$ can be approximated by the ratio of spam messages containing w to the ratio of all spam messages in a training set. This is sufficient to be used by the Bayes's classifier, so we can outline the training and selection process for this model.

Training process

1. Calculate probabilities $P(c)$, $P(x_w = 0|c), P(x_w = 1|c), c \in \{L, S\}$.
2. Using Bayes's formula calculate $P(c|x_w = 0)$ and $P(c|x_w = 1)$.
3. Calculate $\Lambda(x_w), x_w = 0, 1$, calculate $\lambda\dfrac{P(L)}{P(S)}$ and store these values.

Classification process

1. Determine feature vector x_w for message m.
2. Retrieve the stored value $\Lambda(x_w)$.
3. Use Bayes's decision rule to determine class of the message.

Now we need to generalize this classifier to include more features than just the presence of a single word. The simplest way (and a very common one) is to choose a set of most common words $w_1, w_2, \ldots, w_n$ and define the feature vector $x = (x_1, x_2, \ldots, x_n), x_i =$, $x_i = 1$ if the message contains w_i, $x_i = 0$ otherwise.

The problem with this approach is that it requires calculation and storing of all possible values of the feature vector, and there are 2^n such vectors, which is not feasible. A common way to remove this requirement is to assume that the individual components of the vector are independent [1]. This assumption is not formally correct, but in practice it is a good compromise between formal correctness and computational requirements. Because of independence of features:

$$P(x|c) = \prod_{i=1}^{n} P(x_i|c),\ \Lambda(x) = \prod_{i=1}^{n} \Lambda_i(x_i)$$

This classifier is known as Naive Bayesian Classifier due to assumption of independence of features. Training and classification are very simple computationally.

Training process

1. For all $w_i \in W$ calculate and store $\Lambda_i(x_i), x = 0, 1$.
2. Calculate and store $\lambda\dfrac{P(L)}{P(S)}$.

Classification process

1. Determine feature vector x for message m.
2. calculate $\Lambda(x)$ using the stored values $\Lambda_i(x_i)$.
3. Use Naive Bayes's decision rule to determine class of the message.

In terms of word selection for the feature vector, we will speak of the general case when all words from training set are used. This may not be feasible in practice, however, as it will result in a large vector. Usually words that are too common or too rare can be ignored.

Another benefit of naive Bayesian filter is that it is very easy to expand the feature vector to include additional available metadata. In case of e-mails, for example, it would be contents of e-mail headers. It is possible to include additional components either to the calculation of the a-priory probability of the vector or to combine the risk of Bayesian classifier with additional risk calculated from metadata when making a decision.

2.2 k Nearest Neighbors Classifier

Words.

2.3 Artificial Neural Network Classifier

Words.

2.4 SVM Classifier

Words.

2.5 Boosting

Words.

Chapter 3

Advanced Approaches

3.1 Combined Boosting

Words.

3.2 Adaptive Feature Selection

Words.

Conclusion

Words.

References

[1] Konstantin Tretyakov. Machine Learning Techniques in Spam Filtering. Institute of Computer Science, University of Tartu. Data Mining Problem-oriented Seminar, MTAT.03.177, May 2004, pp. 60-79.

[2] Xavier Carreras, Lluis Marquez. Boosting Trees for AntiSpam Email Filtering. TALP Research Center, LSI Department, Universitat Politecnica de Catalunya.

[3] V. Kecman. Learning and Soft Computing. 2001, The MIT Press.