Spoken and Natural Language Understanding Practical Assignment 2

Submitted by Jiahui Dai

April 23, 2025

Accompanying material: Assignment_2.ipynb

1 SMS Spam Detection

Naive Bayes Classifier is used to detect spam in this task [**raschka2014naive**].

1.1 Data Preprocessing Steps

The **bag of words** model is used. It treats each document as a collection of words, disregarding grammar and word order but keeping track of word frequency.

Steps:

1. Tokenisation: breaking down a text corpus into individual tokens
2. Stop words removal: to remove words that are relatively common and uninformative
3. Stemming and Lemmatization:
   (a) Stemming: the transformation a word into its root form
   (b) Lemmatization: to obtain the grammatically correct forms of words
4. N-grams: to group a sequence of n-words together. Unigram model is performed here.

1.2 Experimental Design and Methods

1. Split the data into training and test data
2. Train the Naive Bayes Classifier model on train data
3. Test the model with train data by calculating the probability of spam/ham on a given text, and making a decision if the text is spam or ham.

Class Conditional Probability
To calculate the probability of a word x_i given a class w_j:

$$P(x_i|w_j) = \frac{N_{x_i,w_j}}{N_{w_j}} \tag{1}$$

where

- N_{x_i,w_j}: Number of times word x_i appears in samples from class w_j
- N_{w_j}: Total count of all words in class w_j

To account for words not present in the training data, Laplace smoothing is used to avoid zero probabilities. To calculate the probability of a word x_i given a class w_j:

$$P(x_i|w_j) = \frac{N_{x_i,w_j} + \alpha}{N_{w_j} + \alpha|V|} \tag{2}$$

where

- α: default: 1
- $|V|$: Count of unique vocabulary

To calculate the probability of a message $\mathbf{x}$ given a class w_j:

$$P(\mathbf{x}|w_j) = \prod_{i=1}^{d} P(x_i|w_j) \tag{3}$$

$$= 2^{\sum_{i=1}^{d} \log_2 P(x_i|w_j)} \tag{4}$$

To make a decision:

$$\text{Decision}(X) = \begin{cases} \text{spam} & \text{if } P(w = \text{spam} \mid X) \geq P(w = \text{ham} \mid X) \\ \text{ham} & \text{otherwise} \end{cases} \tag{5}$$

where

$$P(w = \text{spam} \mid X) = \frac{P(X|\text{spam}) \cdot P(\text{spam})}{P(X)} \tag{6}$$

$$P(w = \text{ham} \mid X) = \frac{P(X|\text{ham}) \cdot P(\text{spam})}{P(X)} \tag{7}$$

$$\hat{P}(\text{spam}) = \frac{\text{\# of spam messages in training data}}{\text{\# of all messages in training data}} \tag{8}$$

$$\hat{P}(\text{ham}) = 1 - \hat{P}(\text{spam}) \tag{9}$$

$$P(X) = \sum_j P(X|w_j) \cdot P(w_j) \tag{10}$$

$$= P(X|\text{spam}) \cdot P(\text{spam}) + P(X|\text{ham}) \cdot P(\text{ham}) \tag{11}$$

4. Evaluate the model with its evaluation metrics.

1.3 Hyperparameters

* Laplace smoothing * n-grams

1.4 Evaluation Metric [google_accuracy]

- TP: A spam message is correctly classified as spam
- TN: A ham (non-spam) message is correctly classified as ham
- FP: A ham message is incorrectly classfied as spam
- FN: A spam message is incorrectly classfied as ham

Accuracy is the proportion of all classifications that were correct, whether positive or negative.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

Recall or **true positive rate (TPR)** is the proportion of all actual positives that were classified correctly as positives.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{13}$$

False positive rate (FPR) is the proportion of all actual negatives that were classified incorrectly as positives

$$\text{FPR} = \frac{FP}{FP + TN} \tag{14}$$

Precision is the proportion of all the model's positive classifications that are actually positive.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{15}$$

F_β **Score (F1 Score)** provides a balanced measure of a model's precision (P) and recall (R).

$$F_\beta = (1+\beta^2)\frac{P \cdot R}{\beta^2 \cdot P + R} \tag{16}$$

Matthews correlation coefficient is robustness despite imbalanced dataset.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)\cdot(TP+FN)\cdot(TN+FP)\cdot(TN+FN)}} \tag{17}$$

1.5 Findings

1.5.1 Naive Bayes Classifier Model

1.5.2 TF-IDF [learndatasci_tfidf]

TF-IDF (Term Frequency-Inverse Document Frequency) is used to evaluate how important a word is to a document in a collection or corpus. It helps to weigh terms based on their frequency within a document and their rarity across all documents.

$$TF(t,d) = \frac{\#\text{ terms of } t \text{ appears in document } d}{\text{Total } \#\text{ of terms in document } d} \tag{18}$$

$$IDF(t) = \log\left(\frac{\text{Total } \#\text{ of documents}}{\#\text{ of documents containing } t}\right) \tag{19}$$

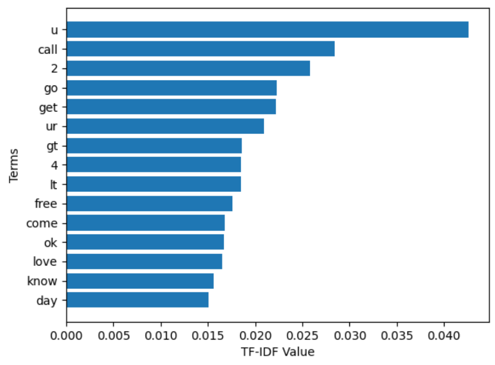
$$TF-IDF(t,d) = TF(t,d) \cdot IDF(t) \tag{20}$$

Term Frequency (TF) measures how frequently a term occurs in a document.

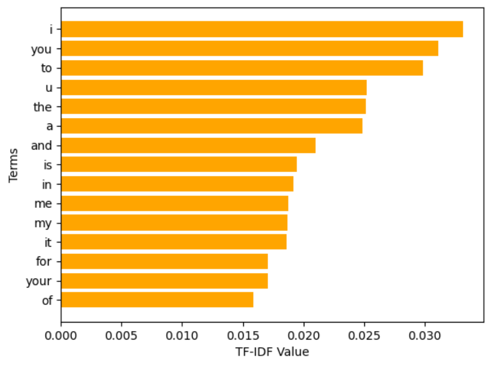
Inverse Document Frequency (IDF) measures the importance of a term across all documents in the corpus.

TF-IDF is the product of TF and IDF. It means that

- If a term appears frequently in a document but also in many documents, the TF will be high but the IDF will be low, lowering the overall score.
- If a term appears frequently in one document but is rare across all documents, the score will be high, emphasizing the importance of that term for the document.



(a) Preprocessing: Tokenisation, stop words removal, stemming



(b) Preprocessing: Tokenisation only

Figure 1: TF-IDF of different preprocessing methods

Figure 1 shows the comparison between the TF-IDF values of top 15 words of models with different preprocessing steps. Model from Figure 1a is be the more "ideal" preprocessing step, with some TF-IDF values being high, emphasising the importance of that term for the document. Compared to model from Figure 1b, the score tends to be on the lower side. This is seen as there are more filler words (stop words not removed), as the words appear frequently within the documents and among all documents. Examples of such stop words are ‘i’, ‘you’ and ‘the’.

The stop words removal from the `ntlk` module may not contain all stop words as some of them are ‘slangs’ or accronyms created by speakers.

1.5.3 Spamminess

There are times where the model encounter words a few times during the learning phase, and it causes the problem of the model to trust blindly based on the information the data provide. To avoid taking unreliable words into account, the spamminess formula is derived:

$$P_s(S|x_i) = \frac{s \cdot P(S) + n \cdot P(S|x_i)}{s+n} \tag{21}$$

where

- $P(S)$: probability of any incoming message to be Spam
- n: number of occurance of this word x_i during the learning phase
- s: strength given to the background information about incoming spam
- $P(S|W)$: spamicity of this word

This formula is the corrected probability for the message to be spam, knowing that it contains a given word.

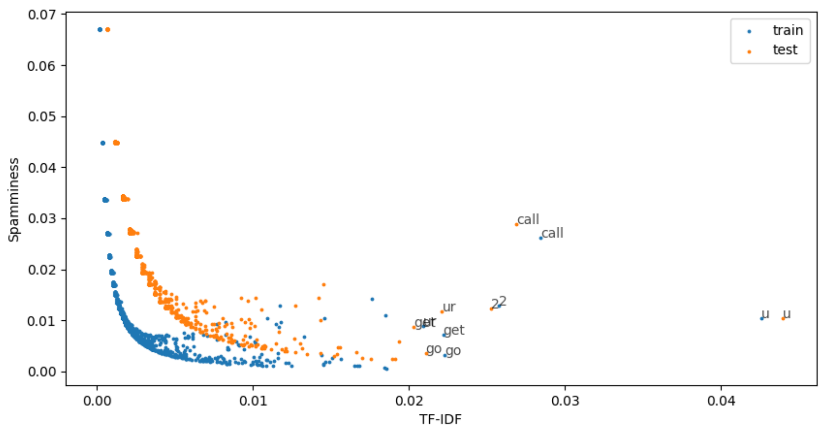


Figure 2: TF-IDF against spamminess

In Figure 2, most words are associated with both low spamminess and TF-IDF scores. This means that the words are more offen classified as ham instead of spam (not particular useful in identifying spam). Its low TF-IDF score also suggests that these words are uninformative or generic, as they are likely to be stopwords not removed during the preprocessing step.

However, there are also words with higher TF-IDF and spamminess score. For example, ‘call’ and ‘u’ are flagged to be an important word to determine spam compared to other words in our trained model.

1.6 Drawbacks [bathula2023limitations]

1. Strong assumption of feature independence. This assumption may not hold true in cases where there is a strong correlation between features.
2. Assumption of the prior probability of each class is known and fixed. However, some prior probability may be unknown or that it changes over time, which affects the classification accuracy.
3. Overfitting if the training data is too small or too many irrelevant features.
4. Curse of dimensionality. If the number of features is large, the probability estimates become less reliable and the algorithm may not be able to capture the underlying structure of the data.

1.7 Potential Improvements

- If test data set has zero frequency issue, apply smoothing techniques like Laplace smoothing to predict the class of test data set. (Attempted)

2 Search Engine

3 Additional Experiments