Kaunas
Faculty

Assignment #2

Basics of Artificial Intelligence

Autorship:
Pablo Mediano Martínez - 77647203P

16/10/2025

YouTube Comment Classification: Spam Detection

Objectives

The goal of this exercise was to understand text analysis using machine learning. The main aims were to:

- Preprocess text data (tokenization, stopword removal, lemmatization).
- Represent text numerically through Bag-of-Words (BoW) and TF-IDF.
- Apply classification algorithms such as Naïve Bayes, Logistic Regression, and SVM.
- Train and evaluate the models using a labelled dataset.
- Measure model performance with accuracy, precision, recall, and F1-score.

1. Dataset

For the experiment we used *YouTube04_Eminem.csv*, part of the YouTube Spam Collection dataset. It contains comments from Eminem videos labelled as spam (True) or non-spam (False).

After loading, the columns and labels were formatted as follows:

```
df = pd.read_csv("youtube04_eminem.csv")
df = df[["content", "class"]]
df = df.rename(columns={"content": "message", "class": "label"})
df["label"] = df["label"].map({True: "spam", False: "ham"})
```

The dataset includes 448 comments: 245 labelled as spam and 203 as ham (legitimate). The balance between classes is reasonable for supervised training.

2. Text Preprocessing

Before training any model, the text must be cleaned and standardized. This experiment used *scikit-learn*'s internal preprocessing tools, which automatically perform the main steps required for text normalization.

The following processes were applied:

- **Tokenization:** Each comment is divided into individual words (tokens).
- **Lowercasing:** All text is converted to lowercase to avoid treating “Spam” and “spam” as different words.

- **Stopword removal:** Common English words such as “the”, “and”, or “is” are filtered out since they carry little meaning.
- **Basic punctuation cleaning:** Punctuation marks and symbols are removed.
- **No stemming or lemmatization:** Although these steps can improve model performance by reducing words to their root form (e.g., “talk”, “talking”), they were not included in this version to keep preprocessing simple.

The dataset was then split into training and testing subsets:

```
X_train, X_test, y_train, y_test = train_test_split(
    *arrays: X, y, test_size=0.2, random_state=42, stratify=y
)
```

This ensures both sets preserve the same ratio of spam and non-spam messages, allowing a fair evaluation.

3. Feature Extraction

Once the text was cleaned and preprocessed, it needed to be transformed into a numerical format that machine learning models can interpret. Two main techniques were used for this purpose:

1. **Bag-of-Words** (**BoW**):
   This method represents each document as a vector of word frequencies. It creates a vocabulary of all unique words in the dataset and counts how many times each word appears in each message. Although simple, it effectively captures word presence and frequency.

2. **TF-IDF (Term Frequency–Inverse Document Frequency):**
   TF-IDF improves on Bag-of-Words by not only considering how often a word appears in a message but also how unique that word is across all messages. Words that appear frequently in spam messages but rarely in others receive higher weights. This helps the model identify distinctive spam-related terms and reduces the influence of common words.

Both methods were applied to the dataset to compare their effect on model performance. TF-IDF produced slightly better results because it emphasizes more informative terms and minimizes the effect of very common words.

The resulting numerical representations were then used as input for the machine learning models described in the next section.

4. Model Training

Three machine learning algorithms were used to classify the comments:

1. **Naïve Bayes (MultinomialNB):**
   A probabilistic model well-suited for word frequency data. It assumes independence between words, making it simple and efficient for text classification.
2. **Logistic Regression:**
   A linear model that estimates the probability of each class (spam or ham) using a logistic function. It often achieves strong performance with TF-IDF features.
3. **Support Vector Machine (SVM):**
   A powerful classifier that seeks the optimal boundary between spam and ham messages in high-dimensional space. It can capture more complex patterns in the data.

Comparing BoW to TF-IDF

Suppose we have three YouTube comments:

Comment	Label
"Great video, really helpful!"	Ham
"Subscribe to my channel"	Spam
"Great video, subscribe now!"	Spam

Bag-of-Words (BoW): Counts how many times each word appears. Example vector for "Great video, really helpful!": [1,1,1,1,0,0,0,0,0].The problem here is that common words like "video" are treated the same as distinctive words.

TF-IDF: Weights words by importance. Frequent words across all comments (e.g., "video", "subscribe") get lower weight, while rare words (e.g., "helpful") get higher weight. The Benefit is that it Highlights distinctive words that help the model distinguish spam from ham.

5. Model Evaluation

Class Distribution

The dataset **YouTube04_Eminem.csv** contains **448 comments**, of which:

- **245** are labelled as *spam*
- **203** are labelled as *ham* (legitimate)

This represents a relatively balanced dataset, suitable for supervised text classification.
For evaluation, **20% of the data (90 samples)** was used as the test set, including:

- **49 spam messages**
- **41 ham messages**

The performance of each model was evaluated using standard metrics: **accuracy**, **precision**, **recall**, **F1-score**, and **support**.

- **Accuracy** measures the overall proportion of correctly classified messages.
- **Precision** quantifies how many messages predicted as spam were truly spam.
- **Recall** (or sensitivity) reflects how many actual spam messages were successfully detected.
- **F1-score** combines precision and recall into a single measure of balance.
- **Support** represents the number of true instances for each class ("ham" or "spam") in the test set.

Modelo	Accuracy	ham Precision	ham Recall	ham F1	spam Precision	spam Recall	spam F1
Naive Bayes (**BoW**)	0.89	0.94	0.80	0.87	0.85	0.96	0.90
Logistic Regression (**TF-IDF**)	0.94	0.93	0.95	0.94	0.96	0.94	0.95
Support Vector Machine (**TF-IDF**)	0.96	0.93	0.98	0.95	0.98	0.94	0.96

- **Naive Bayes (BoW)** achieved good performance, but its recall for *ham* (0.80) shows that it misclassified some legitimate comments as spam. However, it detected most spam messages correctly (recall = 0.96).
- **Logistic Regression (TF-IDF)** provided a more balanced performance between both classes, improving the overall F1-scores thanks to better feature representation using TF-IDF.
- **Support Vector Machine (TF-IDF)** achieved the best results overall with 96% accuracy, successfully identifying almost all messages correctly. It demonstrates strong generalization on unseen data.

Overall, all three models performed well on this dataset, but TF-IDF combined with SVM yielded the highest accuracy and most consistent balance between precision and recall.

6. Results Analysis

The results demonstrate that TF-IDF representations enhance classification accuracy by emphasizing distinctive words. Models like SVM and Logistic Regression benefit from this representation due to their sensitivity to feature scaling.

Possible improvements include:

- Implementing lemmatization or stemming to reduce word variation.
- Incorporating n-grams (e.g., bi-grams or tri-grams) to capture short phrases or patterns.
- Using regularization tuning or cross-validation to optimize hyperparameters.
- Testing deep learning approaches (e.g., LSTM or transformer-based models) for more complex text representations.

Although this project focused on binary text classification for spam detection, the same preprocessing and feature extraction principles can be extended to **topic identification** tasks.

In topic modeling, instead of classifying text into predefined categories (such as spam or ham), the goal is to automatically uncover latent themes within a collection of documents.

Techniques such as **Latent Dirichlet Allocation (LDA)** use probabilistic modeling to identify these hidden topics, while more advanced approaches based on **deep learning**, including **Convolutional Neural Networks (CNNs)**, **Recurrent Neural Networks (RNNs)**, or **transformer-based models**, can capture semantic and contextual relationships at a higher level.

Thus, this spam classification exercise provides a foundation for understanding more complex text analysis applications such as topic discovery and semantic clustering.