Text Classification Model Analysis

Lucy Manalang, Alec Chen

Novemeber 18, 2024

Abstract

In this project, we created two different machine-learning models for text classification and performed two different text classification tasks: sentiment analysis and author identification. Sentiment analysis was performed on 20,000 IMDb examples, and authorship identification was performed on fanfiction.net. Naive Bayes with Lapace smoothing and Logistic Regression Model architectures were evaluated, each with an X f1 score.

Introduction

Text classification is one of the most essential tasks in natural language processing, where a machine is trained to classify different pieces of text with different attributes. This task has significant applications, such as determining sentiment about an online topic or correctly attributing creative works to the correct authors who are their respective authors. Text classification is generally done through supervised machine learning, where the output is a predefined category or label that corresponds to the input text based on the patterns learned from the training data. In this paper, we explore both generative and discriminative models for text classification.

Methods

Data

We performed text classification on two different datasets: the large movie review dataset from Mass et al. [1] and PAN 2018 Cross-Domain Authorship Attribution fan fiction dataset. The large movie review dataset compromises approximately 20,000 reviews in the training set and 5000 reviews in the development set. The fan fiction dataset was organized into individual problems, with each problem consisting of a main text and a specific number of candidates, each accompanied by their own text examples. There were 10 problems in our training dataset and 9 problems in our testing dataset.

Models

We used naive Bayes and logistic regression to perform text classification.

Naive Bayes

The Naive Bayes classifier is a probabilistic model based on Bayes' theorem, which assumes that features are conditionally independent given the class label. The decision rule for classification can be written as:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

where:

- $\hat{y}$ is the predicted class label,
- $\mathcal{Y}$ is the set of possible class labels,
- $P(y)$ is the prior probability of class y,
- $P(x_i \mid y)$ is the likelihood of feature x_i given class y,
- n is the number of features.

The model makes the simplifying assumption of conditional independence:

$$P(x_1, x_2, \ldots, x_n \mid y) = \prod_{i=1}^{n} P(x_i \mid y).$$

Laplace Smoothing

One challenge with Naive Bayes is that the likelihood $P(x_i \mid y)$ may be zero if a feature x_i has not been observed with class y in the training data. To address this, we used **Laplace smoothing**, which adjusts the probabilities by adding a small constant to all counts.

Let N_{y,x_i} be the number of times feature x_i appears in the training data for class y, and N_y be the total number of features observed for class y. With Laplace smoothing, the likelihood is computed as:

$$P(x_i \mid y) = \frac{N_{y,x_i} + 1}{N_y + |V|},$$

where $|V|$ is the size of the vocabulary, i.e., the number of unique feature values.

Implementation Details

To implement Laplace smoothing in the Naive Bayes classifier we:

1. Counted the occurrences of each feature x_i for each class y in the training data.
2. Added 1 to each count N_{y,x_i} to apply Laplace smoothing.
3. Normalized the counts by dividing by $N_y + |V|$ to obtain probabilities.

This smoothing technique ensures that all features have non-zero probabilities, enabling the classifier to handle previously unseen features during prediction effectively.

Logistic Regression

Logistic regression is a linear model for binary classification tasks, where the goal is to model the probability that a given input $\mathbf{x}$ belongs to a particular class $y \in \{0, 1\}$. The probability is modeled using the sigmoid function:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b),$$

where:

- $\mathbf{w}$ is the weight vector,
- b is the bias term,
- $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function.

The decision rule is then given by:

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1 \mid \mathbf{x}) > 0.5, \\ 0 & \text{otherwise.} \end{cases}$$

To enhance the performance of logistic regression on text data, we used the TF-IDF (Term Frequency-Inverse Document Frequency) embeddings to transform raw text into numerical features.

TF-IDF Embedding

TF-IDF is a feature extraction technique commonly used in text classification tasks. It assigns weights to terms in a document based on their frequency and their uniqueness across the corpus. The TF-IDF weight for a term t in a document d is computed as:

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

where:

- $\text{TF}(t, d) = \frac{\text{Number of occurrences of } t \text{ in } d}{\text{Total number of terms in } d}$ is the term frequency,
- $\text{IDF}(t) = \log\left(\frac{N}{1+\text{DF}(t)}\right)$ is the inverse document frequency, with N being the total number of documents and $\text{DF}(t)$ the number of documents containing the term t.

This transformation results in a feature matrix where each document is represented as a vector of TF-IDF weights.

Implementation Details

To build the logistic regression model using TF-IDF embeddings, we implemented the following steps:

1. Preprocessed the text data by tokenizing and lowercasing the input text.
2. Computed the TF-IDF feature matrix using a text vectorizer from the `scikit-learn` library.

3. Split the dataset into training and test sets.

4. Trained a logistic regression model using the TF-IDF feature matrix as input and the class labels as the target variable.

5. Evaluated the model's performance using accuracy, precision, recall, and F1-score metrics on the test set.

Results

IMDb Measures				
Model	Accuracy	Precision	Recall	F1
Baseline	84.4	88.5	82.4	85.3
TF-IDF	50.0	52.23	79.8	85.6

Author-ID Measures				
Model	Accuracy	Precision	Recall	F1
Baseline	7.57	7.57	10.6	8.24
TF-IDF	10.7	10.8	18.9	13.1

We display evaluation metrics: accuracy, precision, recall and f1 for the IMDb and author ID task. In the IMDb task we present an analysis over the entire dataset. In the author ID task we present summary results across all of the problems in this section where results were all macro-averaged results across all candidates in each problem. Each measure was calculated individually and then tabulated in to this table.

It's clear that for the IMDB task, the baseline model performed significantly better than our TF-IDF model, though our evaluation metrics for TF-IDF logistic regression model do not remain consistent across mutliple train / test attempts. In the author ID task, both our models do not perform well but our TF-IDF logistic regression model performs consistently better than the Naive Bayes baseline.

Confusion Matrices for IMDb Dataset

Baseline (IMDb Dataset)

	Positive	Negative
Positive	$TP = 2212$	$FN = 288$
Negative	$FP = 472$	$TN = 2028$

TF-IDF (IMDb Dataset)

	Positive	Negative
Positive	$TP = 2139$	$FN = 361$
Negative	$FP = 1956$	$TN = 544$

The confusion matrices for the IMDb dataset illustrate the stark constract in performance between baseline and TF-IDF model. The baseline model has balanced performance with high

precision and recall for both labels, indicated by relatively low false negatives and false positives. This suggests that the baseline model is effective in classifying without significant bias. In constrast, the TF-IDF shows strong bias towards the positive class. The recall for positive instances is acceptable but the high number of false positives (1956) for the negative class drastically lowers its precision and recall for negative instances. It seems our TF-IDF is unable to generalize well across both labels where our baseline model can.

Discussion

On the IMDb dataset, the TF-IDF model performed poorly compared to the Naive Bayes baseline. We attribute this performance to the dilution of features and handling of rare words. TF-IDF penalizes frequent words which, while helpful in distinguishing between document relevance, may not carry as much weight in sentiment tasks. On the contrary, naive Bayes doesn't penalize word frequency in the same way and provides a more straightforward importance to critical sentiment words.

The limitation of using TF-IDF embeddings for a binary classification task, such as sentiment analysis on the IMDb dataset, harmed our accuracy due to the reduced varaibility in the term weights. With only two classes (positive and negative), the IDF component struggled to create meaningful distinctions between words shared across both classes. For example, a term like "bad" will only have to appear once in a positive and negative review to have no differentiation between their inverse document frequency.

We also suspected implmentation problems with our TF-IDF algorithim and did use the scikit-learn's TF-IDF library with a minor improvement in performance, which can be partly explained by scikit-learn's use of stop-words. As such we stuck with our own implementation of TF-IDF.

Concerningly, our evaluation metrics for IMDb change every run. We suspect this is a result of the random initialization of word embeddings from TF-IDF algorithms. Though, each measure consistently fluctuates around 50 percent. Thus, it is difficult to create a confusion matrix or conclude if the model is better at avoiding false positives or false negatives. However, we can still conclude that the TF-IDF logistic regression model performs worse than the baseline in most cases.

On the Author-ID dataset, the Naive Bayes model performed poorly on all metrics: accuracy, precision, and recall. This suggests the model struggled to learn the distinguishing features of authors in the dataset. Our TF-IDF showed slight improvement from the Naive Bayes baseline with a notable increase of Recall to 18.9% indicating TF-IDF identified more true positives than Naive Bayes. This can be explained by the shorter training documents and more classes to classify, resolving the reduced variability problem identified in the IMDb discussion.

However, overall performance also remained low suggested TF-IDF alone may not provide sufficient discriminatory power for this task. Naive Bayes inability to handle complex feature representations and TF-IDF spare representations may be contributing factors to this poor performance.

Interestingly, in the author ID task, our TF-IDF model performs better with non-English problems, with the model had a relatively high average accuracy of 18.0% on Problems 7 and 9. We believe this is because the problem is non-English and as such, may have more distinct stylistic or linguistic features that could be captured by TF-IDF where the conditional independence assumption of Naive Bayes may break down.[1]

[1]Because we do not know much about the linguistics of French or Italian, it is very difficult to pinpoint what exactly may be the causes of this.

References

[1] Andrew Maas et al. Sentiment analysis dataset. https://ai.stanford.edu/~amaas/data/sentiment/, n.d. Accessed: 2024-11-19.