T. Luke Banaszak

CS410 – Final Project – FA22

Documentation Report

Overview

System Description:

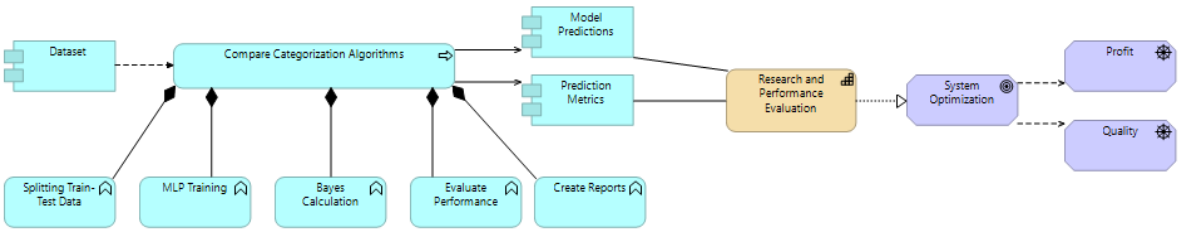
A parametrized text categorization evaluation script; it classifies a user-provided list of texts using two algorithms, Naïve Bayes and multilayer perceptron, producing metrics and reports for easily comparing the algorithms' performances on the dataset.

Implementing these algorithms for text classification requires pre-processing tasks. These tasks change depending on the data structure and specific algorithm resulting in re-implementation for every project. This system provides a single interface to these different algorithms and in a fashion that makes analyzing another dataset easy.

Purpose:

To simplify the workflow for comparing text categorization algorithms' performance on different datasets.

The system provides value by making it easy for analysts and engineers to understand what algorithm they should use for a specific text categorization application. This capability can reduce engineering costs, provide better product quality, and improve customer experience via better categorization performance. The following diagram illustrates the system components, their serving relationship to evaluation capabilities, and the capability's enablement of business drivers like profit and quality.



The system was developed with the use case of predicting what category should be assigned to IT support tickets.

Additional Learning Objectives:

In addition to creating the described system, I also sought to:

- Learn how machine learning algorithms work within the TIS domain; the course content noted that ML methods are common in the domain today, but they were not reviewed in detail
- Identify how (in the current world of ML-focused computing) core TIS concepts are still in use

- Learn usage of TIS and NLP modules that are available in popular data science libraries: SKL, NLTK, Keras and Tensorflow

Background of Project and Modifications to Proposal:

The final version of this project is slightly different from what was outlined in my original proposal, though the learning objectives, knowledge domain, and planned tools are similar.

My original proposal was to create a small system that would accept an IT support ticket as input and predict the associated category using a Bayes classifier. The project's purpose was to demonstrate how valuable text categorization could be within IT service management.

As I began to work on this system, I quickly found that this value stream was already well-documented and proven; it was even mentioned in one of the lectures following the proposal due date. I found multiple articles online where practitioners demonstrated code like what I originally proposed. The authors of these posts used different classification algorithms, though: neural networks, Bayes, SVM, etc. They all used different datasets, though, so there was no way to compare performance.

Rather than simply create another one of these demonstrations, I modified my plan and created this comparison system to understand how the different categorization algorithms perform and compare within the domain of IT support. The resulting system can easily be modified to reach this goal in other knowledge domains with different datasets.

Implementation

Code Overview

The system is implemented as a Jupyter notebook that:

1. Opens a data file containing text and labels
2. Splits data into training/testing sets
3. Tokenizes and TF-IDF weighs text
4. Classifies the test data using Bayes
5. Generates evaluation metrics for the Bayes implementation
6. Trains a MLP network using the training data
7. Predicts using the MLP network
8. Generates evaluation metrics for the MLP network

The notebook imports, and uses functionality from, edited versions of modules from the following repositories.

Google. 2018. "Google Machine Learning Guides." Github. https://github.com/google/eng-edu/blob/main/ml/guides/text_classification/.

Żak, Karol. 2018. "Support Tickets Classification." Github. https://github.com/karolzak/support-tickets-classification.

Some functionality included in these imported modules include the more complex tasks associated with configuring the neural network layers, served as a blueprint for the script's Bayes implementation and reporting, and the tokenization tasks. The dataset used also comes from the second repository.

My code development time centered around building on top of these modules to create a data pipeline that made it easy to submit a dataset for analysis by both algorithms - essentially, making the implementations less "hardcoded" and more parameterized so that they can be used for a variable data file. This required learning and working with TIS features in popular libraries including ScikitLearn, Tensorflow and Keras.

Code Highlights

- Training and test datasets maintain their original frequency distribution when split using SKL's stratification functions
- Data is first split 50% before taking train/test; too much data to train neural net on laptop, otherwise
- The ticket 'title' and 'body' are combined into a single input field for both algorithms
- SKL feature_extraction submodule is used for both algo's tokenization tasks
- Both Bayes and MLP are using unigrams and bi-grams
- Both Bayes and MLP are using TF-IDF weighting
- Bayes method drops stop words during tokenization; MPL includes them but they aren't in neural net because only top 2000 features and included
- The MLP neural net is trained with 20 epochs, but it stops early if performance stops improving

How to Use

The system is presented as a Jupyter notebook. This format demonstrates the system's purpose while separating the components for easy discussion. **Running the system for this demo, then, is as simple as running each cell in the notebook from top to bottom.**

The notebook is currently configured to process a dataset already in the repo containing Helpdesk tickets. Subbing a different dataset into the system, which is its main value proposition, is relatively easy, **but graders should just use the included file for demonstration;** there is still some nuance to the formatting of the input dataset hardcoded into the system. A more robust version of the system, that

can intelligently handle different input formatting, is possible. Global variables initialized early in the script demonstrate where it would be easy to modify the script for different datasets.

The only global variable that can be changed in the demo is "USE_PRELOAD". This tells the script whether to train a new neural network, or just use the serialized version (i.e., the h5 file) included in the repo. Training the network took less than 5 minutes on the author's relatively underpowered laptop, but evaluators may want to avoid the heavy computing and just use the provided model.

After running the script, the user will have two confusion matrices and reports summarizing the accuracy of the two different algorithms on the dataset. The confusion matrix illustrates facts about specific performance: does a certain model perform differently on one specific label? The report includes broad evaluation metrics covered in the course: precision, recall, and F-1.

These results can then be studied depending on the operator's objective. Are they trying to see if a neural network improves performance enough to justify the additional compute time and complexity? Are they having issues with a specific label-algorithm combination and want to see if another algorithm works better?

The next two improvements to the system would be a more user-friendly front end, and a more capable parser for formatting input files. For example, this could be accomplished with a web app, tokenizers and data cleaning tools to allow a user to upload to a GUI a CSV file with, relatively, variable data.

Example Findings from Tool

I developed the system with a dataset of IT Helpdesk tickets containing the request text and associated category. As an example of the system's value, it provided me with the following observations on the performance of these algorithms against what was, essentially, a dataset of 40,000 observations, with document lengths of approximately 150 words, and in a specific business domain.

- The neural network performed better than Bayes with a classification accuracy of about 74% vs 62%
- While the MLP model did perform better, there was significantly more complexity and compute time involved with running it
- Best and worst performing labels were the same between the two algorithms;

References

Google. 2018. "Google Machine Learning Guides." Github. https://github.com/google/eng-edu/blob/main/ml/guides/text_classification/.

Żak, Karol. 2018. "Support Tickets Classification." Github. https://github.com/karolzak/support-tickets-classification.