Machine Learning Task

Submitted by:-

Kandula Aditya Sri Krishna Raghav Chowdary

NITK

Problem Choosen:- EmoInt

Problem Statement:- Develop methods to identify the intensity of emotions in text and submit a technical paper detailing the approaches used.

What is NLP?

Natural Language Processing (NLP). As the name suggests, it’s all about the natural language we use in our everyday communication and deriving insights from the same. Unlike numerical data, textual data is difficult to handle. For one thing, using mathematical models directly on them is not possible.

- As in most machine learning programs, we first need data. You can get textual data from any website like a movie review website, or Amazon product reviews, and so on.

- Here, I'll be using a labeled textual dataset that was provided in Git Repo

- This one's a simple dataset with just four columns, the tweet ID, emotion depicted by the tweet, the author, and the text content of the tweet. We do not necessarily need the author column. Hence we can drop it.

- The dataset has 40,000 tweets in total, labeled into 13 different human sentiments. Our task here is to build a model, such that give a new tweet or text sentence, it can accurately identify which of the emotions (for which it is trained to recognize) it depicts.

- For this tutorial let us consider just two of these sentiments for simplicity, *'happiness'* and *'sadness' (which constitute a total of about 10,000 tweets from the entire sample of data*). We can thus drop rows with all other labels.

- Obviously, we can’t perform math on text and machine learning models are all mathematical models.

- So, how do we convert all these textual data into mathematical data? Remember that we have to take care of countless combinations, special characters, and not to mention, the SMS lingo and slang for which even the dictionary can’t be used for reference.

- First, let’s bring some uniformity to the text by making everything lowercase, removing punctuation, and stop words (like prepositions).

- To gain any proper insight, we need to get all the words to their root form, i.e the variants of a word within the text (for example plural forms, past tense, etc) must all be converted to the base word it represents. This is called lemmatisation. Along with that, I have added code to revert repetition of letters in a word with the assumption that hardly any word has letters repeated more than twice, consecutively. Though not very accurate, it can help in some corrections.

- The next biggest challenge with natural text is dealing with spelling mistakes, especially when it comes to tweets. Apart from that, what do we do about sarcasm or irony in the text? Due to the complexity of dealing with these issues, let us ignore them for now.

- Extending further, one can think of replacing words with their most common synonyms. That could help in building better models. That is being skipped here as well.

- Once you make the text data clean, precise, and error-free, each tweet is represented by a group of keywords. Now, we need to perform 'Feature Extraction', i.e extracting some parameters from the data that can be presented numerically. In this article, we consider two different features, TF-IDF & Count Vectors.
  (Remember, we need numeric data for the math!).

- Split the data into training and testing parts before performing feature extraction.

- **Term Frequency-Inverse Document Frequency (TF-IDF)**: This parameter gives the relative importance of a term in the data and is a measure of how frequently and rarely it appears in the text.

- **Count Vectors:** This is another feature we consider and as the name suggests we transform our tweet into an array having the count of appearances of each word in it. The intuition here is that the text that conveys similar emotions may have the same words repeated over and over again. This is more like the direct approach.

Training Our Models

- With the numerical representations of the tweets ready, we can directly use them as inputs for some classic machine learning models.

- Here, we trained four different machine learning models as demonstrated in the code int Git Repo. We are focusing only on the implementation part. These four methods can, in fact, be used for tackling any kind of classification problem. In our case, we want to classify if a given tweet is a happy tweet or a sad tweet.

- With that being said, I am not going into the details of the inner workings of these algorithms (However, if you are interested to learn more, a simple google search should help you out). For now, being aware of them should sufficient.
  (Also, please note that the syntax for these models is standard.)

First, let us build some models using the TF-IDF features:

Model 1: Multinomial Naive Bayes Classifier
Model 2: Linear SVM
Model 3: logistic regression
Model 4: Random Forest Classifier

The best model had an accuracy of just **54.43%** (**Logistic Regression**) which implies that our model is hardly classifying anything properly. This is no good. This might be because of the complex nature of the textual dataset we are using.

Let’s build models using count vectors features:

Model 1: Multinomial Naive Bayes Classifier
Model 2: Linear SVM
Model 3: Logistic Regression
Model 4: Random Forest Classifier

By using count vectors, we have a significant improvement in performance. The best model, **linear SVM** achieved up to **79.28%** accuracy.

This might be because of the nature of this specific dataset where the emotion of the text is heavily dependent on the presence of some significant adjectives.

- Remember our encodings for the output. ‘0' is for happiness and ‘1’ is for sadness.

- The actual Twitter data can be quite difficult to preprocess. Nevertheless, we can conclude that for normal grammatically correct tweets, our model works pretty well. Using this we could identify the overall view of a group of people whether they are feeling sad or happy correlated to a certain incident or topic in real time. We could also train the model to detect other specific emotions.

- There are several ways to further improve our accuracy like using better preprocessing techniques or using more relatable features. One can also tweak some parameters in the model function to get higher scores.

- And so, our prototype text emotion detection machine is ready!

THANK YOU