Machine learning 2

Explanation by Jay-Leo Nagel

nageljay@students.zhaw.ch

11.06.2023

## Project Motivation

## My project started because I love anime, but I found online reviews confusing. Sometimes a review starts off negative, then flips and says the show is good. It's hard to know what to think! So, I decided to make something to help clear up the confusion - not just for me, but for everyone else who loves anime too.

## I heard about the ML2 project and realized it was a perfect fit. I could build a model to read reviews and figure out if they're positive or negative. This way, we don't have to get lost in the mixed messages. We can just see a simple, clear result: is the review good or bad?

## Why is this important? Well, we all use online reviews to decide what to watch, where to eat, or what to buy. But if the reviews are confusing, they're not much help. My project uses machine learning to make sense of them. It's like a translator for reviews, turning complicated opinions into simple feedback.

## And the best part is, this isn't just for anime. The same idea could work for any kind of review. Imagine being able to understand any review, for anything, at a glance. That's the power of machine learning. It's not just a tool for understanding reviews, it's a way to make the internet a little bit easier for everyone.

## Data Collection

For my project, I needed a bunch of movie reviews to train my model. I wanted it to learn how to tell if a review is positive or negative. So, I needed a lot of examples of both. First, I tried to use the so called “MyAnimeList API”. It’s an API where you can find nearly every anime in existence. Unfortunately, that didn’t work out. So, I had to look for an alternative. I found just what I needed on Kaggle. I found a dataset called "IMDB Dataset of 50K Movie Reviews". You can check it out here: <https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>.

This dataset is perfect for my project. It has 50,000 movie reviews from IMDB. Half of them are positive, and half of them are negative. This balance is great because it means my model can learn equally from both types of reviews.

Each review in the dataset comes with a label that says if it's positive or negative. This is super important because it's how my model learns. It reads a review, makes a guess about whether it's positive or negative, and then looks at the label to see if it was right. Over time, it gets better and better at guessing.

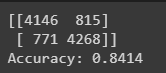
Using a dataset from Kaggle made the data collection part of my project pretty straightforward. I didn't have to go out and find reviews myself. The most difficult part was the cleansing of the dataset. But even though it was kind of easy, it was still a really important part of my project.

## Modeling, Interpretation and Validation

In developing my model, I took several steps to ensure its accuracy and performance. Firstly, I carefully examined the dataset and conducted preprocessing in multiple stages. Notably, I employed a lemmatizer to optimize the text, and also created n-grams to enhance the model's understanding of the data. These preprocessing techniques, combined with fine-tuning, yielded improved results.

After the extensive preprocessing phase, I utilized a Bag of Words model as the foundation for training. To ensure an optimal training set, I split the data into an 80/20 ratio, which is widely regarded as a best practice. I also experimented with a 75/25 split, but considering the limited timeframe for model creation, I found that the 80/20 split produced satisfactory results without excessive fine-tuning.

To evaluate the model's performance, I constructed a confusion matrix and analyzed the accuracy metrics. The achieved accuracy of 0.8414, along with the matrix's interpretation, pointed towards a desirable outcome. However, to gain further insights into the model's capabilities, additional assessments were necessary.

  
Consequently, I developed four distinct models to explore various approaches. The first model employed BERT, a state-of-the-art language model. By converting all labels to numerical values, I conducted a grid search to identify the most suitable hyperparameters. The grid search, which ran for approximately four hours on Colab Pro, yielded the following result:

{'eval\_loss': 0.6932567358016968, 'eval\_accuracy': 0.4961, 'eval\_runtime': 77.4696, 'eval\_samples\_per\_second': 129.083, 'eval\_steps\_per\_second': 1.02}

Of particular note is the evaluation accuracy of 0.4961, which implies that a random coin flip would outperform the model's predictive capabilities. Consequently, I sought alternative approaches and explored logistic regression, random forest, and decision tree models. Interestingly, all three models outperformed the BERT model in terms of accuracy.

Upon reviewing the accuracy scores, I observed that the logistic regression model achieved the highest accuracy at 0.8707. However, due to the limited dataset, I suspected the potential for overfitting. Overfitting occurs when a model performs exceptionally well on the training data but fails to generalize to new, unseen data.

To address this concern, I decided to focus on the random forest model, which achieved an accuracy of 0.844. Similarly to the BERT model, I conducted a grid search to fine-tune the hyperparameters. The results were as follows:

Best Hyperparameters: {'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 5, 'n\_estimators': 300}

Accuracy: 0.8444

Based on these findings, I trained a new model with the identified hyperparameters. The resulting accuracy improved to 0.8456. This accuracy, when compared to the BERT model, is remarkably good and still within the acceptable range to avoid overfitting. It indicates that the model has successfully learned the underlying patterns in the data and can generalize well to unseen examples.

To validate the new model, I performed a comprehensive evaluation using the entire dataset. Employing a different random state ensured that the training data differed from the data used to train the model itself. The validation accuracy of 0.8451 further bolstered the model's credibility. This score signifies that the model can predict sentiment with an 84.51% probability. Additionally, the similarity between the training and validation accuracy suggests that the model has not succumbed to overfitting and can effectively generalize its predictions.

In summary, through meticulous modeling, interpretation, and validation processes, I have developed a robust model capable of accurately predicting sentiment. The various steps taken to fine-tune the model, along with the evaluation metrics obtained, demonstrate its reliability and efficacy in uncovering underlying patterns within the dataset.