Capstone Project:

IMDB Reviews Rating predictor

04.11.2021

Hung Pham

Problem statement and background

Sentiment analysis is one of the earliest problems that come to mind when dealing with NLP. With the world moving to a new digital age with almost anything can be bought and sold on the internet, knowing if a review is positive or negative toward the product is crucial for any business owner. These reviews may contain within it the hint to improve a product or avoid the drawback.

In many cases, the data analyst practitioner will use the rating score to separate all the reviews into good and bad reviews, and just perform the analysis with them. The threshold to decide good from bad reviews differs case by case, but most of the time, it would be enough to learn what influenced users to determine a product good or bad.

But if we want to make it further, say if we have a review and instead of knowing if the review is positive or negative, we want to actually predict the score that reviewer would give to this product. That isn't an easy task, even for us humans (think about when you gave a product a very good review and decide if you want to give it an 8 or a 9).

With all the tools that we have in our toolbox, I decided to push it and see how far we can go in this challenge.

This type of classification problem is indeed a small branch named 'Ordinal Classification' where the target classes have ordinal relationships with each other and we can add a little tweak to our data to take advantage of this extra information. Some advanced techniques were able to reach ~50% precise accuracy and ~80% adjacent accuracy for a 5 classes problem.

Gathering data

I chose IMDb as it is the most popular website for movie information, and I also want to do some extra research on the topic of movies and cinema.

I searched around but could not find any premade data set for user reviews, so I decided it would be a good practice to scrap the data there myself.

The only data I needed for modeling is the reviews' body and their rating, but I decided to also grab the username, date and title of the review to see if I can find any useful information with them

After scraping, I have a small dataset of ~80k reviews from 150 movies

Preparing data for modeling

As my script for scraping puts reviews into separate files for each movie. The first step is to combine them into a master dataframe. As I already set up some failsafe when scraping, there is no missing or duplicated data.

To prepare text data, I build a script to clean (lower case, remove punctuation, number, stop words) and stem it using PorterStemmer. So the data will have a uniform format before transforming.

The class distribution is quite imbalanced as I grabbed the reviews from the extremely good and bad movies. To tackle that, I used imbalance-learn library to strip down the majority classes. The final balanced dataset has ~25k reviews, even for all classes. This number is just ideally for my laptop to handle.

One final step before modeling is using Tfidfvectorizer to transform the text into numerical data. I explained further the main reason behind choosing Tfidf on the notebook.

Modeling

The process for modeling part is quite straight forward: Try different models until satisfied with the best one.

I also define an extra accuracy metric to evaluate the model (a combination of confusion matrix and adjacent accuracy) instead of just using the precise (exact match) accuracy.

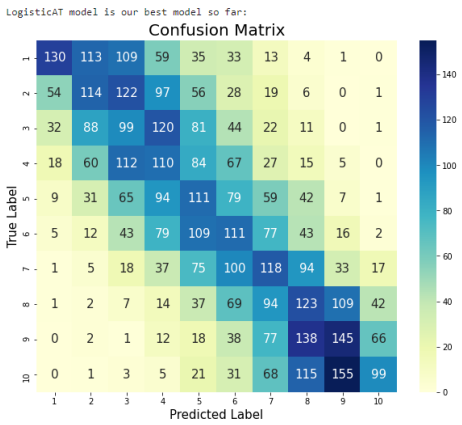
Beside our familiar ML models and Neural network, I found a library specialize for ordinal classification named "MORD" and also tried it out.

Due to the limitation of computing power, I wasn't able to do a gridsearch for all of ML models, instead I ran each model by default first, then picked the winner to optimize.

For neural networks, I chose a regression feedforward model as it is simple and deals well with ordinal classification.

Evaluation and Conclusion

Not too surprisingly, LogisticAT model from MORD library, which is built specifically for this type of problem, easily outperforms other ML models (The accuracy scores are quite close, but if we look at the confusion matrix, it does a much better job).



The Neural network after optimized was able to beat LogisticAT, but only for a small margin interim of score and have a similar confusion matrix. At the same time, NN required way more computational power.

Also, when dumping the NN into a h5 file for use later, the file is massive (~300MB), while LogisticAT's joblib file is merely 70KB.

So considering these aspects, I decided to use LogisticAT as the model of choice to plug in my Streamlit app. The user can paste a link from their favorite movie into the app and see some reviews, as well as the rating predicted by my model.

Future steps

There are many advanced techniques that involve NPL that can push the accuracies much further, like using RNN or CNN in NLP.

Due to the time constraints of this project, I wasn't able to test them all. In the future, I would like to explore them further. A research paper I found mentioned that they can reach about 80% for adjacent accuracy for a 5 ordinal classes, I will try to apply them.