Saood Usmani
Project 4 Report

Introduction and project statement

- Goal: Multi label classification of finance transactions with natural language processing
- I volunteer at a non profit, and one of the things that I help with is classifying miscellaneous finance transactions into different accounts. This can be quite tedious, as there can be hundreds of transactions per month. I have built a machine learning model that can automatically classify these transactions into six different categories that performs very well (.97 accuracy)

Data sources and technologies used

- I used all of the data for the past miscellaneous transactions that are categorized so far. This is around 3000+ data points, which was plenty for my model as my model achieved an accuracy of 97 percent since the patterns are pretty basic. Each data point also had around ten different fields to train on, such as amount, date, donor name, memo(important), etc. I ended up dropping most of these fields as they aren't really relevant to the donation. For example, the balance of the Chase account is definitely not relevant to the donation itself. Another more nuanced example is the donation date. It might theoretically be possible to categorize transactions based on date; say for example, May 1st to May 2nd there was a reconstruction fundraiser. Then, one would expect most of the transactions then to be for the reconstruction fund. However, this would require the model to have knowledge about outside events separate from the transaction itself, which would greatly complicate the model, which is why I decided to drop this column. Generally, most of the columns I dropped were like the first one, where they have no direct or indirect correlation with the transaction, or they exposed sensitive information, detailed below. You can see the exact columns I dropped in cell 3 of my Jupyter Notebook.
- I decided to drop the description column, since that contains very sensitive information such as the type of transaction, the name of the donor, and the transaction ID. However, I wanted to keep the name of the donor as part of the transaction. This is because some donors might donate more towards one fund than the other, and I wanted the model to be able to pick up on these trends. To handle this, I used a secret salt along with a one way hash function to encode donor names. This way, the model can still pick up on these patterns without knowing exactly who the hash was originally.
    - This also opened up a special case of account transfers. For some context, at the end of every month, we transfer the funds out of the misc account to the appropriate accounts based on our categorization. This shows up as a negative balance transaction at the beginning of the next month and this transaction must be categorized. Normally, this would be easy, as you could just look at the account number in the description column and match it with the account number in the chase account. However, due to security and privacy concerns, I did not want to expose these numbers to the model and the VMs. Thus, I added in a preprocessing step that looked for these numbers in the account transfer transactions, and when it found the number it would match it with the appropriate account and add that account to the memo column. This way, the model could

pick up the pattern by looking directly at the memo field, and wouldn't have to look at or know the account numbers at all.

- I used lots of technologies, such as sklearn, pandas, transformers, numpy, torch, joblib, and ChatGPT. I also used React, Vite, and Github Pages for the frontend. I also, if this is relevant, used Chase and Google Sheets as well.

Methods employed

- At a high level, I used natural language processing along with fine tuning for my model.
- First, I would anonymize my miscellaneous spreadsheet using the preprocessing I described above found in cell 3. This would ensure that the spreadsheet was safe to upload to the TACC servers. Then, I combined all the fields I was training on into one text field. Then, I encoded the fund column and also split my data into train, test, and validation sets. Next, I tokenized the text field and built TensorDatasets of my train, test, and validation sets. I then fed this train and test data into a bert model for sequence classification, for fine tuning. I fine tuned over 8 epochs, where I got an accuracy of .94, 100 epochs, where I got an accuracy of .97, and 150 epochs, where I also got an accuracy of .97. Ultimately, I don't think training over more epochs would help, as I seemed to reach a limit of .97 accuracy. Finally, I saved the model, tokenizer, and label encoder so I could use them in my docker image.
- I also spun up a docker image so that the model could be deployed in the cloud. This function would take in an anonymized miscellaneous spreadsheet using similar preprocessing as described earlier. However a key difference was the fund column for each transaction was missing as these transactions hadn't been categorized yet. It would return the same spreadsheet with the fund column filled; effectively, it would categorize each transaction. To do this, I created a flask server that loaded in my model, tokenizer, and label encoder, an inference function within the server that would tokenize and predict each transaction in batches, and an actual post endpoint that would read in the csv, create the text column, call the inference function and set the fund column to the output of the function, and return the csv with the fund column categorized. Finally, I made a simple dockerfile that installed necessary dependencies, copied api.py, the model, tokenizer, and label encoder, and ran the api.py file. I then made an image from this dockerfile after struggling a bit to clean up space, and was able to upload this image to docker hub.
- Finally, I also made a quick website using Vite, React, and Github pages so that a user could use the deployed image endpoint to categorize their csv files online. It works by uploading the uncategorized csv, clicking categorize csv, and then downloading the categorized file. You can find the website here https://saood-usmani.github.io/COE4/

Results

- I was able to achieve 97 percent accuracy with my model on the test set of about 500 transactions, which I was very happy with.
- I was also able to successfully make my docker image that takes in uncategorized transactions and categorizes them, which is also very big since I can actually use this model for practical purposes as well.

    - I actually had to categorize the transactions for April and was able to use the model ato categorize them! It worked fairly well, as it only incorrectly categorized a few(five or so) transactions out of around 150 transactions!
- Finally, with lots of help from Joe, I was able to deploy this image with the endpoint https://fundcategorizer.pods.tacc.tapis.io/categorize. It currently works with curl requests!
    - Example curl request: sudo curl -X POST   -F "file=@testSheet.csv" https://fundcategorizer.pods.tacc.tapis.io/categorize   --output categorized.csv
        - Creates a categorized csv from testSheet.csv
- I have got my final product website up and running so that you can upload an uncategorized csv and get a categorized csv!

ChatGPT usage

- I used ChatGPT extensively for this project to help me figure out what architecture I should use since I was unfamiliar with transformers, help me with preprocessing, help me with debugging, and help me create the docker image. If you want to see exactly how I used it, you can look at this thread here: https://chatgpt.com/share/68191d85-cc50-8013-b565-961989c24889

References

- https://www.youtube.com/watch?v=hn1IkJk24ow