Final Project Writeup

GitHub repo: https://github.com/rayraytheray/Jujube-Fruit

- **Title: Predicting Funding in Indian Startups**
- **Who: Raymond Kao, Efe Alpay, Christopher Williams**

Introduction:

The tech startup scene is fast-paced and exciting, with huge sums of money constantly being invested into businesses around the world. As students with an interest in business, we wanted a deep learning model that could predict how much funding a startup will receive, so that's what we built. While we initially wanted to use a global dataset to create a more general model for predicting startup success, we ended up limiting ourselves to an Indian startup dataset as it had more robust data for what we wanted to achieve.

Methodology

Preprocessing: A large portion of our project was finding and preprocessing the data. The best dataset we found for our task had different CSV files for each month's data. Therefore, we had to combine all of these CSV files into one data frame and remove any noise. Another significant part of our preprocessing was normalizing the funding data. Since funding values went up to millions and were heavily skewed, we needed a way to make them more stable to use with our model. In order to accomplish this, we first applied a log transformation to the funding values, which turned the data into a normal distribution. We then used a MinMaxScaler to ensure the funding values matched the output of our model (between 0 and 1).

Embedding Generation: We used Google's Universal Sentence Encoder, a transformer-based model, to generate an embedding (a 512-dimensional vector) for each short company description.

Model Building: We implemented a dual-structure model consisting of an MLP and a pre-trained Transformer. We used the Transformer model to generate embeddings from short company descriptions, which we fed into the MLP alongside the company's other attributes, such as investors and startup location. The MLP is a simple 4 dense layer model with a sigmoid output layer.

Experiments: We ran a set of ablation experiments where we removed different columns from our data and measured how the model performed without these columns. We also ran our model on a set of different hyperparameters to further optimize performance. The results of these experiments are in the following sections.

Results

- Our model’s training loss drops quickly to around 0.002 (MSE normalized) within 10 epochs. The rate of decrease slows down significantly as the loss approaches 0. We tried increasing the number of epochs, but found little to no increase in performance, suggesting that the model had converged to an optimal solution.
- Our model performs much worse on startups with high funding amounts – it consistently underestimates these values. This is probably because our dataset contains very few startups with these high funding amounts (> $20,000,000), so it lacks sufficient examples to learn the patterns associated with these high-value cases.
- While there were extremely small differences (±0.002) in validation loss and validation MAE over epochs, this is only because the metrics are calculated from normalized values. Funding values ranging from 0 to over 100 million are normalized to 0-1, which means that the “small” decrease in validation loss is actually significant.
- In our ablation experiments, we tried removing various columns from the dataset to see if and how it would affect model performance. While no single column appeared to affect the loss drastically, we found that removing the time information (month and year columns) from the inputs introduced bias. The model overestimated funding amounts compared to before. The ablated model’s validation loss was also greater than the original model’s.
- Our ablated models’ predictions were generally worse compared to our full model as expected.

Challenges

- One of our biggest challenges in this project was getting off the ground with our dataset. It had inconsistent columns, column names, and other formatting issues. It took some time to handle these inconsistencies and eventually merge everything together into a standard format.
- Due to the dataset not being especially large, our model lacked data on startups that received high (> $20 million) amounts of funding. This caused the model to perform poorly on predicting the funding amount that these startups received.
- Initially, our model was not learning as we were normalizing to an output range (log transform of original data) that did not match the output of the model (0-1). We took some time to find a solution, which was normalizing the data further to ensure a 0-1 output.

Reflection

- How do you feel your project ultimately turned out? How did you do relative to your base/target/stretch goals?

We believe that we achieved most of our goals for the project. We created a model that utilized a transformer as well as an MLP to predict, relatively accurately ($50k-$70k error), the amount of funding Indian startups would receive. We managed to create visuals to depict our results and had time to run many ablation experiments. Therefore, we achieved what we say to be our “target” goal. Unfortunately, we didn’t have the time to reach our stretch goal which would’ve been to deploy this model online.

- Did your model work out the way you expected it to?

Our model did work in the way we expected it to – it predicts a funding amount in USD given the other attributes of a startup. However, in our ablation studies, we found that removing the transformer embeddings from model training did not affect the performance as much as we thought it would. We observed very little difference in performance from doing this.

- How did your approach change over time? What kind of pivots did you make, if any? Would you have done it differently if you could do your project over again?

Our approach evolved throughout the project. Initially, we explored using a Graph Neural Network (GNN) architecture because we were interested in modeling the relationships between investors, startups, and funding rounds using categorical data. However, we quickly realized that not only were we unfamiliar with the implementation specifics of GNNs, but we also lacked a clear justification for why GNNs would do a better job than a simple MLP structure, especially given the absence of strong edge features in our dataset.

Instead, we decided to use an MLP with some natural language data. Our new direction focused on using business descriptions as input, which could be encoded into embeddings using a pre-trained language model: Google’s Universal Sentence Encoder. Specifically, we used universal-sentence-encoder-large, which is the transformer-based version. The transformer based version is more computationally expensive, but offers better performance than the default version which uses deep averaging networks. One constraint we faced was the availability of datasets that included meaningful textual descriptions. We settled on an Indian startup dataset, not because of an initial interest in that specific market, but because it was the only dataset we found that included consistent short descriptions of the startups. With more time or better familiarity with GNNs, we might have been able to extract value from the graph structure of startup ecosystems, especially if we had access to richer relational data. Additionally, we would have made an earlier push to curate or build a more diverse dataset with a global scope and better text quality.

- What could you improve further if you had more time?

Our initial intention was to use a global dataset to generalize the model's predictive power across various startup contexts, but limitations in data availability and preprocessing time led us to restrict the scope. A broader, more diverse dataset encompassing startups from multiple countries, industries, and ideally with more data on the founders, would allow the model to better capture patterns across different geographies. Additionally, the text descriptions provided still were not perfect.

- What are your biggest takeaways from this project/What did you learn?

Our biggest takeaway was that our model can really only be as good as our data, no matter how much refinement and testing we do. We also learned that the simplest solution is sometimes the best solution – although we didn't get to implement the graph neural network approach, we may have become lost in the complexities of GNNs and ended up with a less-finished product.