Predict Chain

Matthew Pisano, William Hawkins

Spring 2023

1 Introduction

Throughout our project, we have built upon, or have related to, the ideas of many other projects and papers. Some of these relations were explicit, such as our work with, and utilization of, various types of neural network. Others were implicit, like the relation that our project has to the many other implementations of blockchain-based predictive projects. It is important to acknowledge the influences that other works have has on our project, and it is equally important to analyze the similarities between our project and the other papers that we have referenced here. Through this, we hope to better understand the perspective of others on similar topics to our project and hopefully improve upon our work and techniques in the future.

2 Related Work

2.1 Long Short-Term Memory Networks

Long Short-Term Memory Networks[3] is one of the most influential, and useful, types of network that we work with in our project. These networks introduce several important improvements on classical backpropagation-through-time recurrent neural networks. One of the most notable contributions of LSTMs is their resolution of the 'exploding and vanishing gradients' problem. In RNNs, it is common, and often disruptive, to observe error signals either grow to extreme values or disappear entirely when updating the weights of the model. This either leads to extreme unpredictability in the next value that weights will take on, or it leads to the weights not being updated at all. In both cases, the effectiveness of the model is greatly impacted. To help remedy this issue, the authors propose an architecture that enforces a 'constant ... error flow through internal states' inside the model.

Before this paper, it was well known that traditional RNNs performed very poorly when given long time lagged datasets. In this data, the time window that the input data is set in is several time steps ahead of the window that contains the target prediction. This was so pronounced that, in another paper[2], the authors showed that other contemporary RNN algorithms can be outperformed my random guessing. To show that their new architecture is superior in areas that RNNs fall short in, the authors devise a series of six experiments. Through these experiments, hey show that LSTMs can outperform RNNs on long, minimal time lag data. Through their papers, the authors show that their new LSTMs can perform well in both short and long time lagged situations.

Relation to Our Work

The work of Hochreiter and Schmidhuber has been incredibly useful in our development of PredictChain. As part of our goal, our project is designed to accept a wide variety of datasets and is also designed to train models on a wide range of parameters. Part of this promise of variety is our models' abilities to work with both short and long time lagged datasets. While some of our models fall short in some areas, such as RNNs in a long lagged context, others are able to excell in those areas, such as our LSTM and GRU type models. Through our incorporation of the authors' work, we are able to provide a large range of models for a wide variety of use cases. This helps the project to achieve its goal of greater accessibility in the field of predictive modeling.

2.2 Recurrent Sequence Modeling

One of the core aspects of our project is our usage of various types of neural network for predictive modeling. Most of our neural networks have some recurrent capability, specifically our RNN, LSTM, and GRU model types. Additionally, the data that we suggest that these models be trained on is sequential data, exactly what these types of model were designed for. One paper that aligns especially closely with this component of our project is *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*[1].

Within this paper, the authors aim to evaluate the relative performance of RNNs, LSTM networks, and GRU networks. Similar to our use case, they perform this evaluation using sequential signal data, similar in nature to the stock market dataset that we use in our example. Primarily, they focus on the LSTM and GRU variants of recurrent neural networks, rather than vanilla RNNs themselves. As part of their evaluation, they compare the performance of LSTM unit, GRU unit, and tanh unit RNNs on modeling sequences of polyphonic music and speech signals.

As part of their evaluation, they utilize three publicly available music datasets along with two internal speech datasets from Ubisoft. The design of their experiment is to compare the negative log probabilities of the models after they are given 20 consecutive samples of audio data and are then tasked to predict the next 10 consecutive samples. In the results that they gathered, they concluded that, overall, the GRU unit outperformed the other two evaluated models. These results are amplified in the Ubisoft datasets, where GRU unis significantly outperform the others in one of the test cases and have similar performance to LSTMs in the other. Additionally, in the Ubisoft Dataset B, the negative log-likelihood of the GRU model continues to shrink in a super-linear manner, whereas the other models become asymptotic.

Relation to Our Work

This paper both confirmed some of the initial assumptions that we had about the models and changed some others. In the design of the project, we had initially marked GRU models to be better than all the other models. We denoted this perceived difference in output quality through the constant multiplier we applied to each of their *model_complexity* values. After our analysis of this paper, we ran several tests to see if this paper's results also applied to our project. In our tests, we noticed something very similar to the paper. GRUs and LSTMs has more similar performance than we initially thought. Meanwhile, vanilla RNNs lagged behind. Based off of this evidence, we decided to update these values to better reflect the experimental performance of these models.

Table 1: *model_complexity* Values of Our Models

Model	**Initial Value**	**Final value**
MLP	1	1
RNN	1.5	1.7
LSTM	2	2.2
GRU	2.5	2.4

In our changes, we amplified the values related to all RNN-based networks. We also brought the complexities of the LSTM and GRU networks closer together. This change was made to reflect their relative performances to each other and to more traditional RNNs. Overall, this paper has been important for our project as it has helped us to reflect upon our models and how to best evaluate them.

2.3 Stock Predictions

This next paper is closely aligned with the example dataset that we are using for PredictChain. They focus on using big data to train predictive machine learning models to forecast the short-term future behavior of stocks within the Chinese stock market [4]. Within their paper, they utilize several techniques designed to cut down on the noise that is inherent to the stock market to allow for better predictions. In their paper, they focus mainly on three methods that they use to accomplish this goal:

- Creating a new dataset, gathered primarily from the *Tushare* API

- Performing feature engineering on their dataset to ensure that it can be better used by their model
- Customizing a Long Short-Term Memory model for the actual predictions

For their dataset, they had gathered the past performance of 3558 stocks from the Chinese stock market over a period of two years. Each entry in their dataset contains many attributes including the daily price of a given stock and the ID of that stock. In order to gather this data, they utilized the *Tushare* API, along with web-scraping from *Sina Finance* and the *SWS Research* website.

As part of their feature engineering, they utilized three primary techniques. First, they applied feature extension to their dataset. Through the use of this technique, they added additional meta-attributes to the entries, such as polarizing or calculating the fluctuation percentage. Adding this meta-data to the dataset helps to give the model a more complex picture of the stock and how it performs over time. Next, they eliminated some features by using the Recursive Feature Elimination (RFE) algorithm. This algorithm evaluated each feature based on the degree to which that feature influenced the stock's performance. By eliminating unnecessary features, they allow their model to pay more attention to the most influential features, improving its predictive capabilities.

Finally, they developed their models. They used a two layered LSTM model, with the model only having an input layer and an output layer. As for their output, they kept the output as simple as possible to make sure the model was focussed only on predicting the movement of the stock. The model either outputted a one for the stock going up at a given time step, or a zero if the model believed the stock would go down. Their hope is that this simplicity will also simplify the complexity of the information that investors will have to interact with.

Relation to Our Work

As mentioned previously, the general purpose of this work mirrors part of the functionality of our project. The main difference here, being that it is much more focused on the predictive aspect. Along with their LSTM model, they test several other models on their dataset as a comparison. Out of these models, the LSTM performed the best, with the multi-layered perceptron and the other, non-neural network, algorithms falling behind. Their observed performance matches our initial assumption and the modifications we had made after reviewing [1].

One notable feature that this paper has that our project implicitly lacks is their complex preprocessing of the dataset. At present, PredictChain leaves the majority of dataset preprocessing up to the uploading user. That is, our project does not perform any feature engineering before submitting the dataset to models. A future improvement that we could make to our project is to give users the option of having feature extension or elimination automatically applied to their dataset after submission. This change may result in yet better performance of our models after training on user-submitted datasets.

3 Closing Thoughts

After researching and reading these relevant papers, we were able to gain valuable insight into projects similar to our own, implement relevant features, and take note of some features that could be added in future revisions of PredictChain. The knowledge that we have gained from these papers falls broadly into two categories: improvements that we could make to our usage of artificial intelligence and improvements that we could make to our usage of blockchain.

The papers that had focused on the AI component helped us to re-revaluate how we rate our models and how we can best improve the data processing and training aspects of the project. [3] and [1] helped us to better understand how we should rank the potential effectiveness of our models through our usage of their *model_complexity* values. Specifically, [3] encouraged us to expand the degree to which users could customize the time lag that is present in the dataset upon model training. [4] helped us to better understand some of the future improvements that we could make to both our dataset preprocessing and model training.

As a summary, this chary illustrates the impact that each of the papers had on our work and our knowledge:

Table 2: Impact of Each Paper

Paper	**Notable Impact**
Long Short-Term Memory[3]	Encouraged us to expand our support for forcing time lag in datasets and expanding the degree to which users could customize the lookback windows
Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling[1]	Helping us to better understand the relative performances ofRNNs, LSTMs, and GRU networks
Short-term stock market price trend prediction using a comprehensive deep learning system[4]	Inspiring possible future improvements to our dataset preprocessing and model training

References

[1] Chung, Gulcehre, Cho, and Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Neural and Evolutionary Computing*, 2014.

[2] Hochreiter and Schmidhuber. Bridging long time lags by weight guessing and "long short term memory". 1996.

[3] Hochreiter and Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[4] Shen and Shafiq. Short-term stock market price trend prediction using a comprehensive deep learning system. *Journal of Big Data*, 2020.