Predicting The Success of Crowdfunding

Chenchen Pan
MS&E
Stanford University
cpan2@stanford.edu

Yiwen Guo
ICME
Stanford University
yguo46@stanford.edu

Yan Chen
Department of Statistics
Stanford University
ychen16@stanford.edu

Abstract

Crowdfunding platform like Kickstarter, where entrepreneurs and artists seek for support from a large number of contributors, has become prominent over the past decade. Better understanding and more accurate prediction of the success of a project can help both contributors and creators make better use of their resources. Previous works applied traditional machine learning methods, such as SVM and Random Forest, with only categorical and numerical features, such as goal and duration, or only textual features, such as project description and keywords. To our knowledge, we are the first work to apply deep neural networks with all three types of features. Using a dataset of 100K+ crowdfunding projects, our model achieves 72.78% accuracy on test set, which is significantly better than the baselines and the previous works. We also show that the trained model can help us understand the success and failure of crowdfunding projects better.

1 Introduction

Crowdfunding is the practice of funding a project by raising money from a lot of people via the Internet. Within a variety of such online platforms, Kickstarter is an American company that allows people to crowdfund for their projects across the world. People with new ideas post details about their projects on this website, in which they specify the amount of money they wish to raise, the deadline, and so on. If people can know how likely they will achieve success given their initial ideas, they can then modify the details of their projects to improve the likelihood of success. Therefore, in this project, given some key features of a project, we want to predict the probability of its success. More specifically, the input features are of three types: text(such as description and key words), continuous variables(such as the amount of money a person wants to raise), and categorical variables(such as country and currency). We then use random forest, logistic regression, and recurrent neural network to output a predicted probability of success.

2 Related work

As crowdfunding becomes more and more popular, many researchers have tried to conduct various experiments on understanding the dynamics behind it. For instance, Mollick took a holistic view and proposed that personal networks, underlying project quality, and that geography were the most important factors in determining the success of the crowdfunding [8]. Mitra and Gilbert took a different approach and focused on analyzing the language in crowdfunding. By studying a huge corpus of texts presented in 45K projects, they found that phrases following certain principles such as reciprocity, scarcity, and social identity increased the likelihood of success [1][7].

More recently, researchers have used machine learning algorithms to gain more insight of crowdfunding. For example, Sawhney, Tran, and Tuason built a binary classifier that predicted the success of

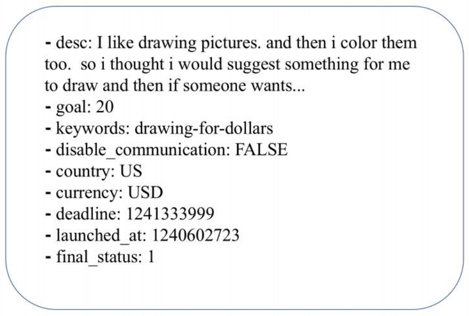
CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

a campaign by analyzing campaign content, linguistic features, and meta-information at the time of its inception through sentiment analysis and latent Dirichlet allocation [9]. Desai, Gupta, and Truong also focused on the role of language. They trained a series of discriminative models for binary success/failure prediction of projects with increasing complexity of linguistic features, showing that successful project pitches were, on average, more emotive, thoughtful and colloquial [3]. Hussain, Greenberg, Chen, and Etter applied a number of binary classification models such as random forests, logistic regression, k-nearest neighbors, and support vector classifier, and obtained reasonable results [2][4][5][6]. In addition to predicting the success of crowdfunding, Zhou looked at the relationship between pledge amount and category, and concluded that the size of projects was proportional to the amount of pledges creators received [10].

Indeed, these works propose a variety of efficient methods that we can use to predict the success of crowdfunding. However, we notice that none of them implement neural network. In this project, we aim to apply different neural network architectures to solving this problem.

3 Dataset and Features

We use a dataset from Kaggle [11], which has 108,129 examples in total. Note that among the 100K examples the number of success is about 30% and that of failure is about 70%. For instance, one positive example looks like the following:



- desc: I like drawing pictures. and then i color them too. so i thought i would suggest something for me to draw and then if someone wants...
- goal: 20
- keywords: drawing-for-dollars
- disable_communication: FALSE
- country: US
- currency: USD
- deadline: 1241333999
- launched_at: 1240602723
- final_status: 1

Figure 1: A positive example

Here, "desc" is the description of the project, "goal" is the amount of money the creator wants to raise, and "disable_communication" is whether the creator is willing to communicate with funders. Notice that "launched_at" and "deadline" are in unix time. We then split the data into 90% training set, 5% dev set, and 5% test set. The input features are : description, goal, keywords, disable_communication, country, currency, and the time difference between when the project is launched and the deadline. Note that we drop the number of backers since we are interested in predicting the success of a project at its inception time.

4 Methods

4.1 Data Preprocessing

To begin with, we turned "country", "currency", and "disable_communication" columns into categorical variables having values from 0 to the number of class minus one. Then, we added a new column called duration, which was computed by subtracting the date when the project was launched from the deadline date. Next, we normalized all continuous variables including "goal" and "duration". After that, we used different approach to encode and embed text features. For instance, we used one-hot vector representation in our baseline models, and used pre-trained 300-dimensional GloVe vectors in our RNN model.

4.2 Baseline Models

We ran random forest classifier with 100 trees and shallow neural networks as our baseline models. Since the implementation of random forest was straight-forward, we would devote much of the space here for the shallow neural network. First, we used one-hot vector representation for each of the text features, namely "desc" and "keywords". Then we did a word embedding for each of the two with a vocabulary size of 50 words and flattened the outputs into two vectors. For continuous and categorical variables, we did a linear layer. Then we concatenated these three outputs and passed them as inputs to a neural network with one hidden layer. In this layer, we first chose ReLU and then sigmoid for the activation function. In addition to the architecture described above, we also used L2 regularization and Adam optimization. For the cost function, we chose binary cross-entropy loss, which is defined as

$$
\begin{aligned}
J &= \frac{1}{m}\sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L}\left\|w^{[l]}\right\|_F^2 \\
&= -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)})] + \frac{\lambda}{2m}\sum_{l=1}^{L}\left\|w^{[l]}\right\|_F^2
\end{aligned}
$$

4.3 Recurrent Neural Network

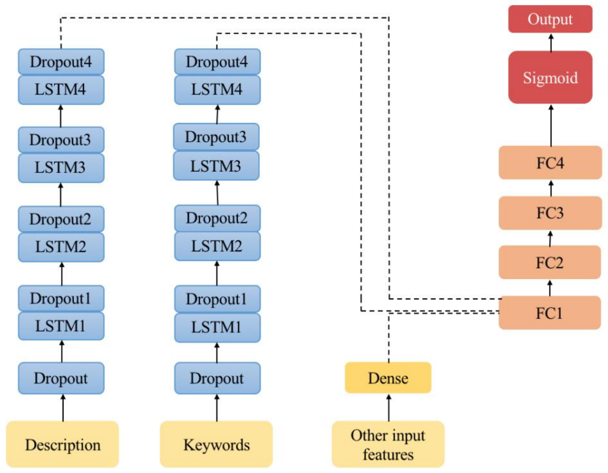


Figure 2: RNN Model Architecture

A standard neural network may not be the best when we work with sequence data such as audio and texts. First, inputs and outputs can be different lengths in different examples. Second, a standard neural network doesn't share features learned across different positions of text. Therefore, we used RNN as our model architecture. More specifically, different from the baseline models, we implemented a 300-dimensional GloVe vectors to represent words with a size of 840 billion vocabularies. After that, we did a dropout layer with a probability 0.5. Then we repeated LSTM and dropout four times. For other input features, we did a dense layer. Next we concatenated these three outputs and fed them into a fully connected layer. Finally, we used the sigmoid activation function for our last layer (see Figure 2). For regularization, we used both dropout and early stopping.

5 Experiments

5.1 Hyperparameter Tuning

We used random search for tuning hyperparameters. A list of hyperparameters and their associated range is summarized in the table below (see Table 1).

Hyperparameters	Range
Dropout rate	0-0.5
Number of LSTM layers	2-7
# of hidden units in one LSTM layer	64,128,256
Learning rate	0.0001-0.01

Table 1: Setting of hyperparameters tuning

5.2 Results

The primary metric we use is accuracy. The first part of Table 2 shows our best-performing hyper-parameters. Then the second part of Table 2 summarizes the result. The RNN model achieves the highest accuracy of 72.42%. Figure 3 and Figure 4 illustrate the accuracy and cross-entropy loss of our RNN model against the number of epochs.

Hyperparameters	Value
LSTM state size	64
Dropout rate	0.485
FC layer output sizes	64
Learning rate	0.000145
Epoch	50

Models	Dev Accuracy
Random Forest Classification	70.00%
Shallow Neural Network	69.35%
RNN Model	**72.42%**

Table 2: Experiments Results

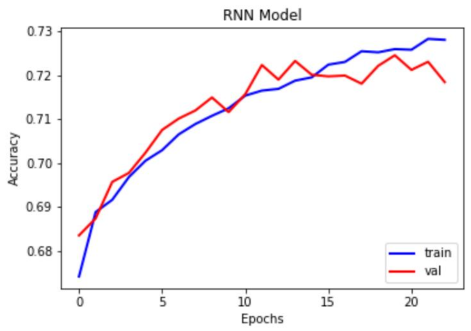


Figure 3: Accuracy of our RNN model against the number of epochs

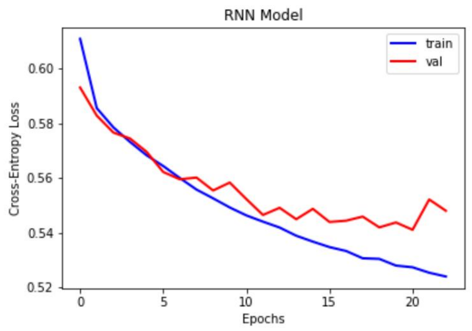


Figure 4: Cross-entropy loss of our RNN model against the number of epochs

5.3 Error Analysis

Although our approach significantly outperforms previous method, the room for improvement seems big. After all, the accuracy for the prediction is still low and only around 72.5%. We randomly sampled 100 projects that our model did not predict correctly, and categorized the errors. About 26% of errors are in fact due to dataset issues and are not real mistakes. This includes label error (6%) and incomplete information (20% e.g., there are some missing content with quotation marks in project's description). 29% of the errors due to the fact that the model sometimes only uses partial information for predictions. It tends to predict success or failure based on certain words ("game", "documentary" and "film" for success and "app", "food" and "mobile" for failure) without considering the global information. The rest 45% are model errors. Note that this task is quite challenging for human as well because a human can only get around 50% correct.

Figure 5 is the confusion matrix on test set. There are 3324 true negatives, 611 true positives, 383 false positives, and 1089 false negatives.

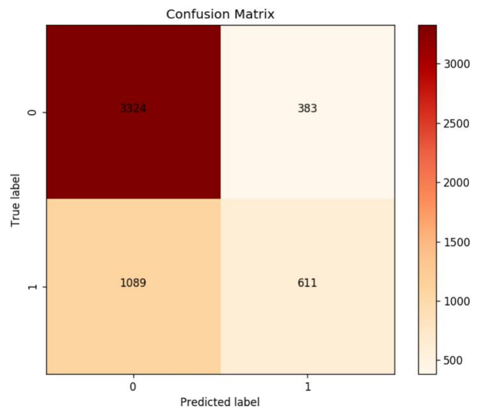


Figure 5: Confusion matrix

6 Future Work

For data issue: We can try to collect project samples with more complete description information. Or we may just remove the samples whose description information is severely missed (but this will lead to the loss of information and prediction power). In addition, we can spend some time to correct the mismatched labels. Besides, we can also try to collect the image and video information and include it in our features. Because intuitively they also probably impact the final status of a project. Also, we'll apply our model on the datasets used in previous work to make a comparison of the performance.

For model issues: We notice that there are quite a lot of wrongly predicted examples which tend to only use local information of the descriptions or the keywords. So in future, we plan to apply more sequence models like bi-directional LSTM to process the text features. Also we plan to apply batch normalization, which may also help in our future modeling. We may also try to weight the loss functions regarding different misclassification cases.

7 Contributions

At the beginning, Yan Chen, Yiwen Guo, and Chenchen Pan worked together to preprocess data. Then, Yan mainly focused on neural network modeling. Chenchen contributed to running all baseline models and designing RNN architecture. Yiwen did literature review and organized the structure of this report.

8 Github Repository

https://github.com/chenchenpan/Predict-The-Success-of-Crowdfunding

References

[1] Tim Althoff, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. How to ask for a favor: A case study on the success of altruistic requests. In *ICWSM*, 2014.

[2] Kevin Chen, Brock Jones, Isaac Kim, and Brooklyn Schlamp. Kickpredict: Predicting kickstarter success. Technical report, Technical report, California Institute of Technology, 2013.

[3] Nihit Desai, Raghav Gupta, and Karen Truong. Plead or pitch? the role of language in kickstarter project success, 2015.

[4] Vincent Etter, Matthias Grossglauser, and Patrick Thiran. Launch hard or go home!: predicting the success of kickstarter campaigns. In *Proceedings of the first ACM conference on Online social networks*, pages 177–182. ACM, 2013.

[5] Michael D Greenberg, Bryan Pardo, Karthic Hariharan, and Elizabeth Gerber. Crowdfunding support tools: predicting success & failure. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 1815–1820. ACM, 2013.

[6] Nida Hussain, Kareem Kamel, and Archana Radhakrishna. Predicting the success of kickstarter campaigns.

[7] Tanushree Mitra and Eric Gilbert. The language that gets people to give: Phrases that predict success on kickstarter. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 49–61. ACM, 2014.

[8] Ethan Mollick. The dynamics of crowdfunding: An exploratory study. *Journal of business venturing*, 29(1):1–16, 2014.

[9] Kartik Sawhney, Caelin Tran, and Ramon Tuason. Using language to predict kickstarter success.

[10] Peter Haochen Zhou. Predicting the success of kickstarter campaigns. 2017.

[11] https://www.kaggle.com/iamsajanbhagat/kickstarter/data