# NLP Research - Kickstarter Fundraising Blurbs

Yuan Zhou

## Abstract

In this research, we created and trained different Neural Network models (1D-CNN, DNN, LSTM, and GRU) with variant text encoding technologies and dropout rates, aiming to find the best model that could predict the possibility of successful fundraising when given the blurbs of a Kickstarter project.

## 1. Introduction

Many start-ups are using Kickstarter to raise money in the hope to prototype their ideas from scratch. We know an illustrative and persuasive expression of ideas is critical to fundraising success. We want to reveal the correlation between them in real-life projects and help start-ups succeed in their early stage by conveying attractive ideas. In our research, we would apply many projects' fundraising history data in Kickstarter to our designed Neural Networks to generate a machine learning model that predicts the possibility of success for any given blurb of a project. This model could help start-ups get more confidence in prototype validations on kickstarter.com.

## 2. Literature Reviews

Deep Learning has been widely used in various NLP problems, especially in recent years. In their paper, Socher R. et al. introduced CNN, a breakthrough framework for computer vision and image recognition, with its variations, such as 1D-CNN, RNN, and Ranking Network work and how they are applied to solve NLP problems[1].

Strubell, E., Ganesh, et al. put some effort into its standardization: they introduced an

NLP-Cube: an end-to-end Natural Language Processing framework, which performs sentence splitting, tokenization, compound word expansion, lemmatization, tagging, and parsing. The learning model was based entirely on recurrent neural networks. [2]

## 3. Methodology

### 3.1 Dataset Overviews

The whole dataset has three columns: project-ID, a blurb of the project, and fundraising status ('successful' or 'failed'). There are 214,118 records in total, and the distribution of successful and failed projects is about 1:1 as we count.

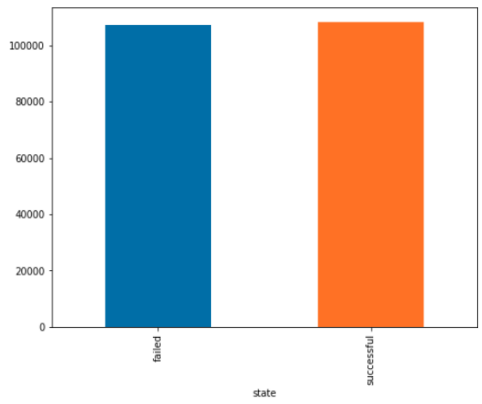



Fig-1 Distribution of failed and successful records

### 3.2 Data preprocessing and encoding

We first reserved 20% of the whole dataset as a test dataset. As for the training dataset (about 170,000 records), we used the first 130,000 for training and the last 40,000 for validation. We also converted all labels to integers (successful: 0, failed: 1).

As for text preprocessing, we removed stop words and completed word stemming and tokenized the text in each row based on the dictionary generated from all training documents. We also did truncating/padding in the final numeric sequences generation to align the sequences' length in all rows.

### 3.3 Neural Network Modeling

We used a factorial design with alternative core Neural Network (NN) layers to train, evaluate, and select the best model amongst all candidate NNs. In each model, we kept most factors the same: number of layers, number of units in each layer, hyperparameters, optimizations, max epochs, early stoppers, etc.; only the NN

algorithm is one of the four candidates: DNN, 1D-CNN, LSTM, and GRU. A typical NN model without dropout layers is shown in Fig-2:

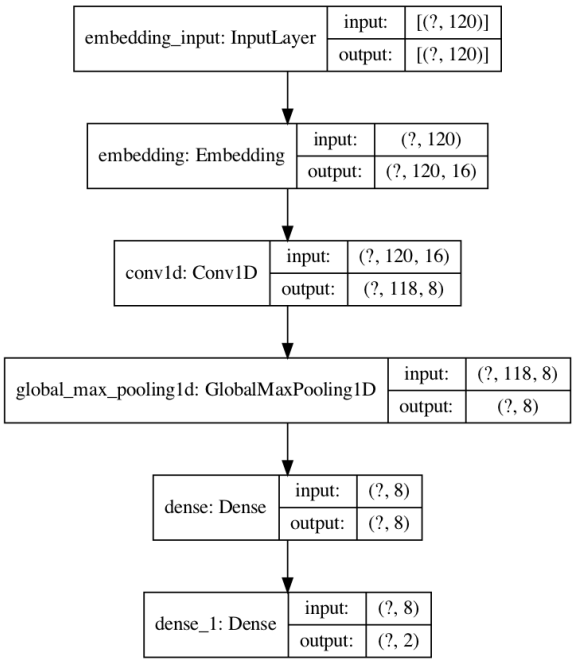



Fig-2 1D-CNN model without dropout layers

**3.4 Combat overfitting problem**

We already reserved part of the training set as validation as a dataset used to fight the overfitting problem. We added an early stopping checking in every model training process, which monitors the val_accuracy from each epoch. Another attempt to fight overfitting was to apply dropout layers between NN layers. We selected a set of dropout rates: 0, 0.2, 0.4, 0.5, and used them to each NN model to evaluate their outcome.

## 4. Results & Evaluation

We have conducted a series of experiments and recorded all results in Appx-Table-1. We will use the following sections to evaluate every significant variance we introduced and draw a conclusion for the best model.

**4.1 Neural Network Algorithms Evaluations**

We trained four different NN algorithms: DNN, 1D-CNN, LSTM, and GRU. We use one column where *encoding* = word_embedding as an example, to compare the results of the best F-1 score for each algorithm amongst different *dropout_rate*: ***LSTM*** (*F1*=0.6703, *dropout_rate*=0.5) > ***GRU*** (*F1*=0.668, *dropout_rate* =0.4) > ***DNN*** (*F1*=0.6659, *dropout_rate*=0.5) > ***1D-CNN*** (*F1*=0.6621, *dropout_rate*=0.2). As the comparison results shown, all models performance are very close (0.6621 ~

0.6703), and LSTM is slightly ahead of others.

Another noticeable result is training time per epoch: DNN (3s) < 1D-CNN (4s) < LSTM (23s) < GRU (29s). DNN and 1D-CNN are at the same efficiency level, and LSTM and GRU are much slower (about 8-9 times) than DNN and 1D-CNN.

And when encoding (word_embedding) and dropout rates (0.5) are all set the same, we could tell from Appx-Fig-1 that the *AUC_ROC* ranking among the four candidates is: LSTM > GRU > DNN > 1D-CNN.

Given the analysis above, we concluded the LSTM is the most suitable NN algorithm for this classification problem.

#### 4.2 Encoding Algorithms Evaluations

The encoding candidates to evaluate and compare are *one-hot encoding* and *word embedding*. From the result, we could conclude that the word embedding is a superior encoding option than one-hot, as all of its F-1 score, test_accuracy, and training time per epoch are better in every model when other configurations are the same.

#### 4.3 Overfitting Evaluations

On average, the total training epochs are about 3 to 4 times for all models, which is much smaller than the configuration where max_epochs=100, thanks to the early-stopping monitors. On the other hand, the dropout rate setup did help improve the final performance result. As we can see in 4.1, all the best F-1 scores for each model are produced when dropout layers are introduced.

#### 4.4 Optimization

The model we started as a basis for optimization was the LSTM model, whose encoding = word_embedding and dropout_rate = 0.5. We attempted a few optimizations that included:

- increasing/decreasing units within the LSTM

- Stacking more LSTM layers in sequence
- Adding L2 regularization to LSTM layers
- Fine-tuning dropout rate and early stopping patience

We applied them individually and combined, but the final F-1 score and test accuracy were still around 0.67. Thus, we believed the experiments' models have fully exploited the correlation between the blurb and the final result.

## 5. Conclusion

From the experiment and results evaluation, we could conclude: we could build a Neural Network model that could predict the success/failure of a fundraising campaign in Kickstarter when given its short description. The best model we got is about 67% accuracy, which means the blurb and fundraising results are related (significantly greater than 50%). We also made a few attempts to improve the performance but gained no obvious outcome. Thus, we concluded that a project blurb has a significant effect on the final fundraising result; and to further explore the possibility of increasing chances of success, we need to collect other essential factors for creating more columns in the dataset. For example, the full-text description, the photographs, video introductions, and most importantly, the idea's quality.

## References

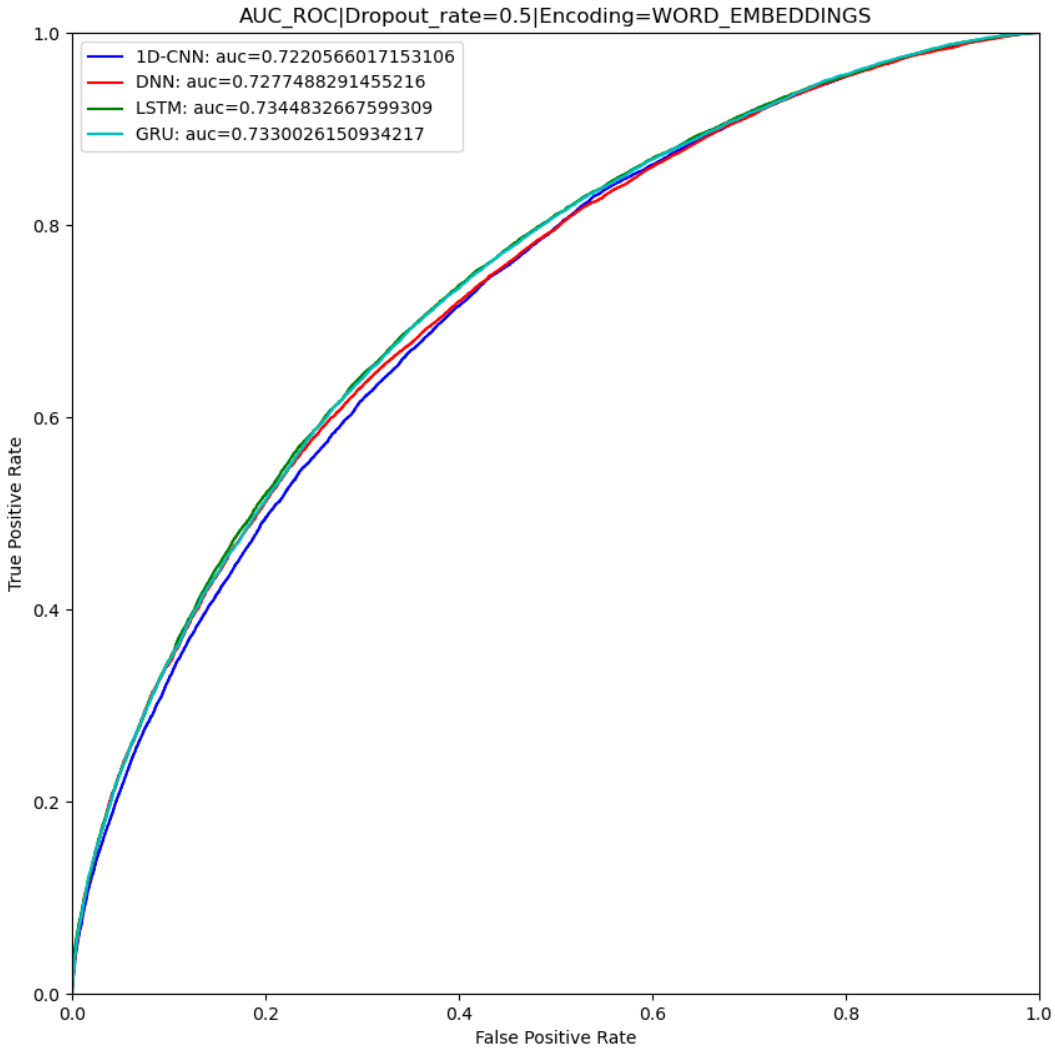
[1] Socher, Richard, Yoshua Bengio, and Christopher D. Manning. "Deep learning for NLP (without magic)." In Tutorial Abstracts of ACL 2012, pp. 5-5. 2012.

[2] Strubell, Emma, Ananya Ganesh, and Andrew McCallum. "Energy and policy considerations for deep learning in NLP." arXiv preprint arXiv:1906.02243 (2019).

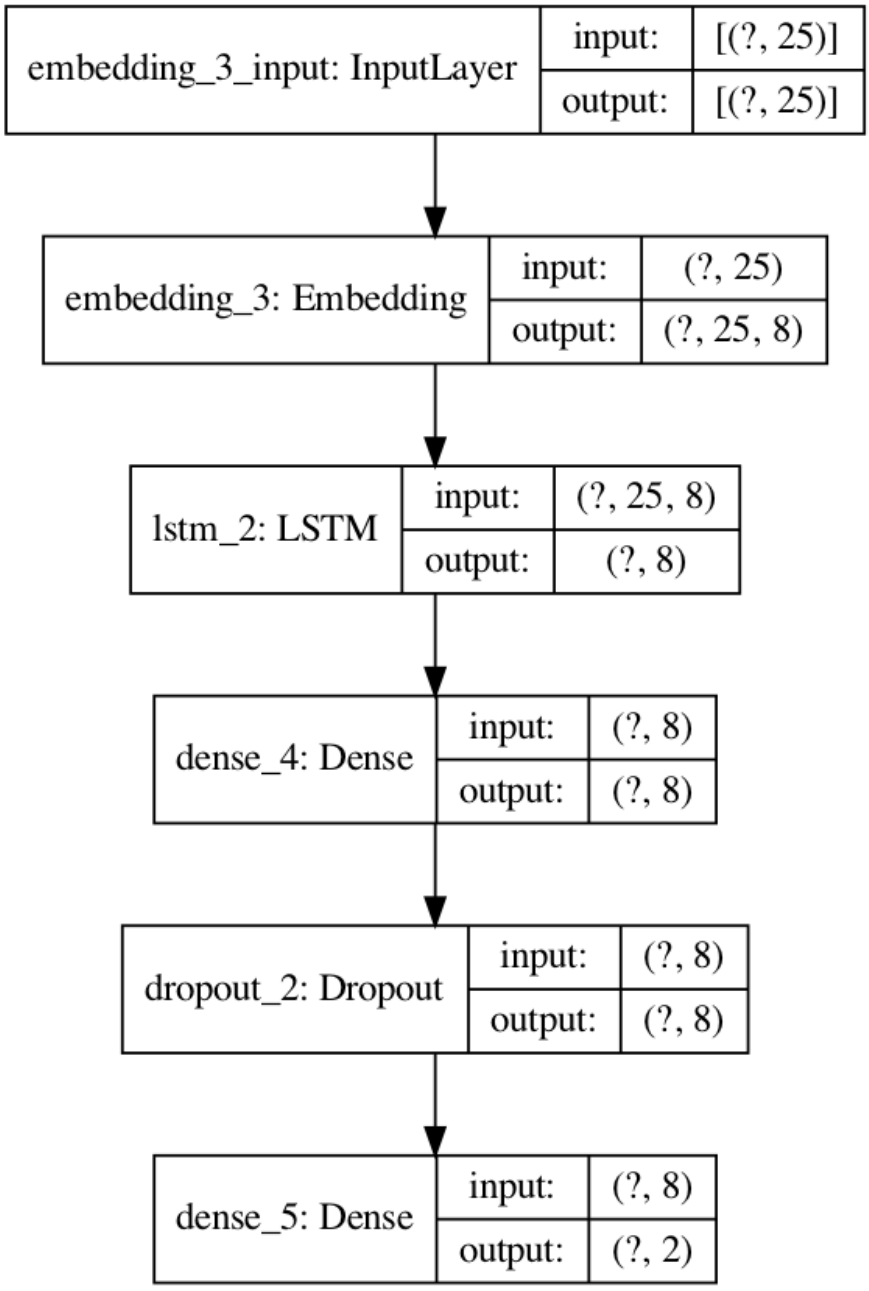
## Appendix

| Model | 1D-CNN | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Encoding | Word Embedding | | | | One-hot Encoding | | | |
| Dropout rate | 0 | 0.2 | 0.4 | 0.5 | 0 | 0.2 | 0.4 | 0.5 |
| F-1 score | 0.6561 | 0.6621 | 0.6234 | 0.6599 | 0.6213 | 0.6222 | 0.6222 | 0.6211 |
| Test accuracy | 0.6566 | 0.6626 | 0.6388 | 0.6601 | 0.6229 | 0.6224 | 0.6226 | 0.622 |
| Training time per epoch | 4s | 4s | 4s | 4s | 60s | 61s | 63s | 60s |
| | | | | | | | | |
| Model | DNN | | | | | | | |
| Encoding | Word Embedding | | | | One-hot Encoding | | | |
| Dropout rate | 0 | 0.2 | 0.4 | 0.5 | 0 | 0.2 | 0.4 | 0.5 |
| F-1 score | 0.6604 | 0.6647 | 0.6654 | 0.6659 | 0.6088 | 0.6063 | 0.6065 | 0.6043 |
| Test accuracy | 0.6609 | 0.6648 | 0.6654 | 0.6659 | 0.6091 | 0.6082 | 0.6069 | 0.6057 |
| Training time per epoch | 3s | 3s | 3s | 3s | 32s | 32s | 33s | 33s |
| | | | | | | | | |
| Model | LSTM | | | | | | | |
| Encoding | Word Embedding | | | | One-hot Encoding | | | |
| Dropout rate | 0 | 0.2 | 0.4 | 0.5 | 0 | 0.2 | 0.4 | 0.5 |
| F-1 score | 0.6625 | 0.67 | 0.6461 | 0.6703 | 0.6065 | 0.6018 | 0.6187 | 0.6232 |
| Test accuracy | 0.6665 | 0.6707 | 0.6561 | 0.6703 | 0.618 | 0.6151 | 0.6227 | 0.6239 |
| Training time per epoch | 23s | 22s | 22s | 24s | 56s | 55s | 56s | 56s |
| | | | | | | | | |
| Model | GRU | | | | | | | |
| Encoding | Word Embedding | | | | One-hot Encoding | | | |
| Dropout rate | 0 | 0.2 | 0.4 | 0.5 | 0 | 0.2 | 0.4 | 0.5 |
| F-1 score | 0.6634 | 0.6632 | 0.668 | 0.6612 | 0.612 | 0.6001 | 0.5982 | 0.5994 |
| Test accuracy | 0.6669 | 0.6663 | 0.6679 | 0.6649 | 0.6207 | 0.615 | 0.6142 | 0.6137 |
| Training time per epoch | 29s | 26s | 28s | 26s | 56s | 56s | 56s | 56s |

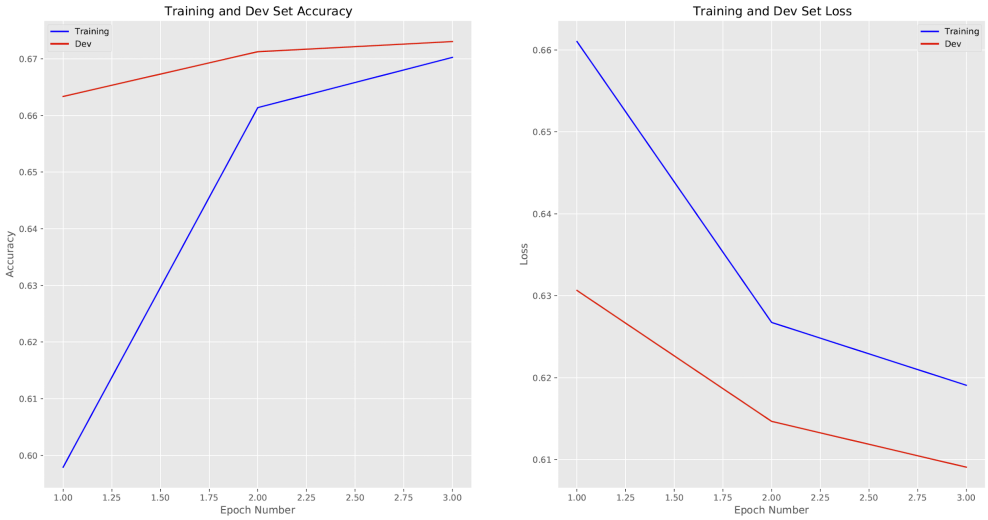
Appx-Table-1 Neural Network models training evaluation results



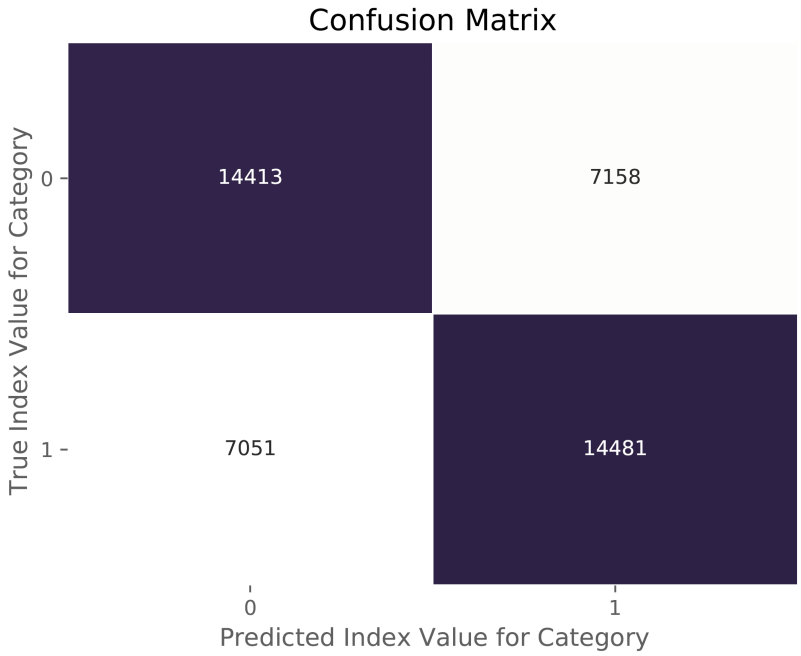

Appx-Fig-1 ROC curve and AUC_ROC for all for NN algorithms




Appx-Fig-2 The architecture of the LSTM model with dropout layers




Appx-Fig-3 Training history for LSTM model with dropout=0.5 and encoding=word_embedding




Appx-Fig-4 Confusion Matrix LSTM model with dropout=0.5 and encoding=word_embedding