Prediction of Kickstarter success with deep neural networks

Emile Jaspar
Tilburg University
Cognitive Science and Artificial Intelligence
e.j.o.a.jaspar@tilburguniversity.edu

Abstract

Crowdfunding platforms like Kickstarter have risen in their prominence since they seek great support for entrepreneurs of new projects. However, only one-third of the projects reach their goal. A better understanding and more accurate prediction of the success can be of great importance for the creators and backers at the start.
The aim of this paper is to explore the performance of deep neural networks on a Kickstarter dataset, compared with machine learning techniques.
We compared the performances of a k-Nearest Neighbors with a Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM) and a Convolutional Neural Network (ConvNet). The best classifier overall is the ConvNet, LSTM performs best on accuracy and the MLP performs best on precision and recall.
This paper has shown that the performance on predicting the success of a Kickstart project through deep learning methods are better than machine learning techniques. Although this difference is slight.

1 Introduction

Kickstarter[1] is an enormous global community built around creativity and full of ambitious, innovative, and imaginative projects that are brought to life by direct support of others, backers. Before launching a campaign, the creator sets a funding goal and a deadline. Until the deadline, it is possible to pledge money towards the project in order to bring it to life. The support by the backers can be rewarded through the involvement in a process of the project, unique rewards offered by the project or unique experiences to the project.

Since only 36% of the Kickstart projects reach their goal, is it of great importance for the creators that they are notified about the probability of success early on. This makes it possible to intervene on time. A project that seems not to be classified positive might need a new strategy in order to be successful. Meanwhile, the projects with a good prospect could already start working on their project or prepare for a soft launch. At the same time, the backers can take advantage of these classification predictions as well. As soon as the prospect is negative, could the engagement of friends and family by backing the project lead to a better expectation. When the probability of success is high, backers could consider withdrawing their pledge or adjust it, with the idea that the project will succeed [Etters et al., 2013].

There have been several studies done in predicting the success of a campaign. These studies mainly made use of machine learning classifiers. Etters et al. [2013] studied the prediction of the success of campaigns based on two features: the time-series of money pledged and social attributes. The combination of these two features led to an increased accuracy, in the very early stage of the campaign (88%). Sawhney et al. [2016] did something similar and investigated if it is possible to use the information at the beginning of a project to predict its success, without making use of dependent data or data external to the Kickstarter project itself. Their classifier achieved an accuracy above 70%. In addition, Mollick [2014] offered exploratory insights into several determinant features of success and failure of crowdsourcing projects. He analyzed the leverages of these features and other campaign characteristics on the outcome of the project. Lastly, Greenberg et al. [2013] tried to predict the success of a crowdfunding project at the time of the project launch. They reached an accuracy of 68%.

This experiment will analyze the possibility of using deep neural networks to predict the outcome of a campaign. The dataset comes from Kaggle and needed to be pre-processed. In this small experiment, we hope to contribute more with the predictions of deep neural networks to the success of a project, than machine learning techniques. So, the research question will be as follow:

"Can we predict the outcome of a Kickstart project better with deep neural networks than with machine learning techniques and to what extent?"

[1] https://www.kickstarter.com/help/taxes

In Section 2, we describe the architectures and methods of the different deep neural networks. We then give a plain overview of the methods and set-up of this experiment in Section 3; explaining the data, the experiment, hyperparameters, and the evaluation criterion. In Section 4 we present the results, which will be discussed in the 5th Section. Finally, we conclude and discuss future work in Section 6.

2. Deep learning architecture and methods

In this paper we will make use of three different deep learning methods which will be built with Keras[2]: the Multilayer Perceptron (MLP), the Long Short-Term Memory (LSTM) and the Convolutional Neural Network (ConvNet) to answer the research question.

The MLP makes use of an Adam optimizer. This optimizer is referred to a method for stochastic optimization for only first-order gradients with little memory requirement [Kingma & Ba, 2014]. As activation, we made use of the "Relu".

The LSTM also makes use of the Adam optimizer and a Dropout layer. The Dropout will be used to prevent units from co-adapting too much. The dropout significantly reduces overfitting and improves performance [Srivastava et al., 2014].

The dataset has a fixed length and therefore we used a one-dimensional ConvNet. The ConvNet for this report contains several methods which are already mentioned in the other two deep neural networks namely the Adam optimizer and the Dropout method. Besides that, the ConvNet also consists of two pooling methods; first is max pooling, which reduces the size of hidden layers by an integer multiplicative factor α [Graham, 2014]. Second, is the global average pooling. This is more meaningful and interpretable because it mediates between feature maps and categories. Besides that, global average pooling is a structural regularizer and by that prevents overfitting for the overall structure [Lin, Chen & Yan, 2013]. As activation, we made us of the "Relu".

3. Experimental method and set-up

In comparison to related studies, this research will be small. We made use of a preprocessed dataset and most of the algorithms that are used for the predictions are basic.

3.1 Data (statistical overview)

In this research, we used a dataset which was offered by Kaggle. Mickaël Mouillé pre-processed the data which he got from the Kickstarter platform. Mouillé his data consists of 378662 rows and 15 columns, but we removed nine columns and 8208 rows. The nine columns were unnecessary, and the 8208 rows were missing values. Two of these columns were transformed together into one column that represents the difference in days between the launched day and the deadline. An overview of the statistical properties is shown in Table 1 and Table 2. Four columns consisted of words and needed to be replaced by numbers, whereby each unique word got his own number. Through making use of a RobustScaler[3] for the input variables we are able to do reduce the effects of outliers and causes a better performance of the algorithms. On the contrary, we have the output with three classes. This means that we deal with a dataset with multi-classes and therefore we made use of LabelBinarizer[4].

We trained the neural networks to predict three different classes; successful (36%), canceled (11%) and failed (53%). Most literature predicted only two classes, but the chance that a project will be canceled is existing. So, we decided to include this class in this research. Besides that, the dataset is imbalanced, and in order to solve that we first made use of class weights in the neural network to minimalize the influence on the outcome through this imparity.

Before feeding the data into the algorithm of a ConvNet or the LSTM, we need to reshape it into three dimensions. After that, we split the dataset in a training set and test set, respectively 66,66% and 33,33%. While we trained the data, we separated this dataset in a 20% validation and 80% training set.

	main_category	currency	state	country
unique	15	14	3	23
top	Film & Video	USD	failed	US
freq	62282	289837	197719	289671

Table 1: Global statistics of our dataset of Kickstarter campaigns. We show information about all the columns which contain strings.

	usd pledged	usd_goal_real	deadline_launched
mean	7076.437966	45076.376313	33.406512
std	78822.582579	1128264.968841	60.689139
min	0.000000	0.010000	0.000000
max	20338986.270000	166361390.710000	14866.000000

Table 2: Global statistics of our dataset of Kickstarter campaigns. We show information about all the columns which contain numeric values.

3.2 Experiment

Since the aim of this research is a comparison between the performance of a machine learning technique and of deep learning models, we started with building a machine learning model first. We decided to build k-Nearest Neighbors

[2] An open-source neural-network library written in Python, link: https://keras.io/

[3] Transforms the feature vector by subtracting the median and then dividing by the interquartile range (75% value—25% value)

[4] A utility class to help create a label indicator matrix from a list of multi-class labels:

(kNN) classifier, which tries to identify *k* records in the training set that are similar to a new record that we wish to classify. When we want to classify the new record into a class, we assign the new record to the predominant class among these neighbors. We build the kNN by making use of the scikit-learn[5] package.

The first layer of the MLP has 34 neurons. The second layer returns a shape of 300 and the third shape of 150 and the last layer returns it into three classes.

For the LSTM we made use of a simple model, with a many-to-one structure. In this structure, a sequence of multiple steps as input is mapped to class or quantity one output value. The LSTM consist of three layers with LSTM. The first two layers return a sequence of 32 and the last one returns one single vector of 32 dimensions. Before the last LSTM layer, we made use of a Dropout layer to make sure that we will not overfit.

After that, we adjust a default one-dimensional Convent, which is often used for time sequences. The first layer defines a filter (or feature detector) of height four (kernel size). If we would use only one filter the neural network would learn one single feature. This could be not enough for this dataset, so we first used 200 filters. The result of the first layer will be used by the second layer to define 200 different filters to be trained on this level and will be turned into a matrix of 3 x 200. To decrease the complexity of the total output and prevent overfitting we will make use of max pooling. We used a size of three which means that the size of the output is one-third of the input. Then the third and fourth layers learned more levels features. The output after these layers is 1 x 260. To be certain that the algorithm will not overfit we add a global average pooling layer so that each feature detector has one weight. The dropout layer is being used to allocate within a range of 10% zero weights to the neurons. In the hope that the network will not be too sensitive to react to smaller variations in data. The last layer reduces the height of 260 to 3 classes. All the layers with activation function are making use of the Relu. Except for the last layer which uses a SoftMax function, the output will represent the probability for each class [Ackermann, 2018].

Since all networks are ready, we will train it by making use of callbacks. This is a set of functions which makes it possible to view the internal states and statistics of the model during training. The combination of the method ModelCheckpoint and EarlyStopping will prevent the network to overfit and makes it possible to visualize the training process in a graph.

After running the algorithm, the accuracy for the neural networks did not increase anymore regardless of the chosen parameters. After revising these changes, we encountered that the accuracy was not reliable enough to measure the performance of this algorithm. So, we added two other criteria: the confusion matrix and an overview of the precision, recall, and F1-score. After analyzing these measurements, we found that the contribution of one major class resulted in high accuracy. This is also the reason why it lacked in increasing accuracy. The minor class has not been predicted by the model.

The effect of adding class weights to the algorithm seemed not to be powerful enough. This has led to a more serious intervention, we could choose between three methods: undersampling, oversampling and SMOTE. The last one is the most commonly used preprocess method, which employs the k-nearest neighbor technique for over-sampling the minority class [Chawla et al., 2002]. After all, the results were much more reliable than before.

3.3 Hyper-parameter settings

Most hyper-parameters settings are already explained in the paragraph above regarding the structure of the models. For the time being the deep learning models used the default settings of the optimizers, except the MLP. The learning rate for this algorithm is 0.0001, both betas are 0. For the other models, only the epochs and batch size for each model has been optimized. Choosing the hyperparameter for kNN was the easiest one, in the early changes we noticed that the number of neighbors for predicting the best prediction resulted in three k's. The deep neural networks took much more time. After an exhaustive search on a wide range of batch sizes and evaluating the results with the test set, we found for all the neural networks the best hyperparameters. The optimal parameters each deep neural network is visualized in Table 3.

	MLP	*LSTM*	*ConvNet*
batch size	29	200	150
epochs	22	20	33

Table 3: Overview of the optimal parameters for each deep neural network for this research.

3.4 Evaluation criterion

As explained in the experiment part we only used the accuracy first. The simple MLP in the experiment will be used to set a baseline for the other deep learning models. The result of training the MLP is the accuracy of 75.70%. After discovering that we cannot make conclusions only based accuracy, we also returned a confusion matrix and an overview of the Precision, Recall and F1-score per class. For defining the best model, we need to look at the Recall score for the classes "canceled" and "failures" and at the Precision score for the successful class. Overall, we need to set a minimum performance of 60% for each class independent of which performance measure will be used in the confusion matrix.

4. Results

[5] A free software machine learning library for the Python programming language, link: https://scikit-learn.org/stable/

By choosing the best parameters for each deep neural network, through examining many different values and evaluating the performances with the validation set, we first put all the confusion matrices together. By looking at the confusion matrix, we can have an idea of how each classification model is performing. A confusion matrix consists of four different elements;

- True Positives (TP) are the classes that are predicted positive and that are true. The diagonal of each confusion matrix.
- False Positives (FP) are the classes that are predicted as positive but are negative. The sum of values in the corresponding row of a class, excluding the TP
- True Negatives (TN) are the classes that are predicted as negative and these are true. The sum of values in the corresponding column of a class, excluding the TP.
- False Negatives (FN) are the classes that are predicted as negative but are positive. The sum of all columns and rows excluding that class his column and row.

With these elements, we can explain more in detail what the results are for each classifier and how they perform. In Figure 1 we see that the diagonals for each algorithm are the darkest. This means that if the algorithms predict positive the actual class is positive, so we have a lot of true positives. We can notice a huge difference in the true positives of the class 'canceled' and 'successful' of the MLP. Besides that, we can see another remarkable result. For each algorithm the class 'failed' does not have many false positives nor false negatives and a lot of true positives. It means that for each algorithm the predictions for this class are very accurate.

Since we have a clear overview of all confusion matrices, we can dive deeper into the performance of each. We will make use of two calculation methods such as Precision and Recall.

With precision, we can define the percentage of the results which are relevant. In simple terms, high precision means that an algorithm returns substantially more relevant results than irrelevant ones. It can be calculated with the mathematical equation shown in Figure 2.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Figure 2: The equation of precision

In this paper, we will also make use of the recall (also known as sensitivity) in order to define the percentage of total relevant results correctly classified. This means that if the algorithm has a high recall it returns most of the relevant results. The equation is shown in Figure 3.

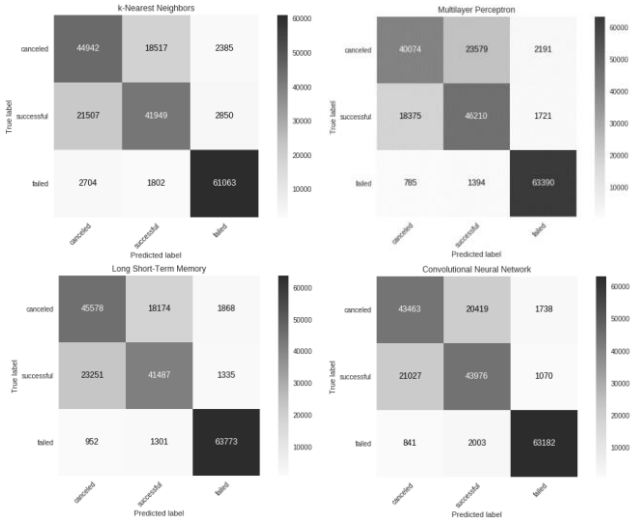

Figure 1: An overview of all the confusion matrices which are made through different algorithms. The darkness of the diagonals indicates the number of predictions that are predicted correctly. Every square excluded from the diagonal indicates the amounts of predictions that are predicted wrong.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Figure 3: The equation of recall

In Figure 4 we see a clear overview of all the precisions for each class and each algorithm. We want the amount of precision for the classes 'canceled' and 'failed' as high as possible. When the precision for these classes is low, many projects will be predicted as 'successful' while the actual label should be 'canceled' or 'failed'. The wrong outcome of these predictions will have a big influence on the results of a project. A low score on precision for the class 'successful' does not have that negative costs. As you can notice the MLP scores the best, followed by the ConvNet.

Regarding the results of the recall, we want the opposite. The consequences of a false negative in the class of 'successful' can be devastating. A low recall for 'successful' means that the algorithm predicts often a 'failed' or 'canceled' case as 'successful'. It is therefore important that the score of recall for 'successful' is high. By this, we can ensure that we don't miss any 'successful' classes in the prediction. If we look at Figure 5, we can see that the high true positives of 'successful' of the MLP result in the highest recall for 'successful', but the lowest for 'canceled'.

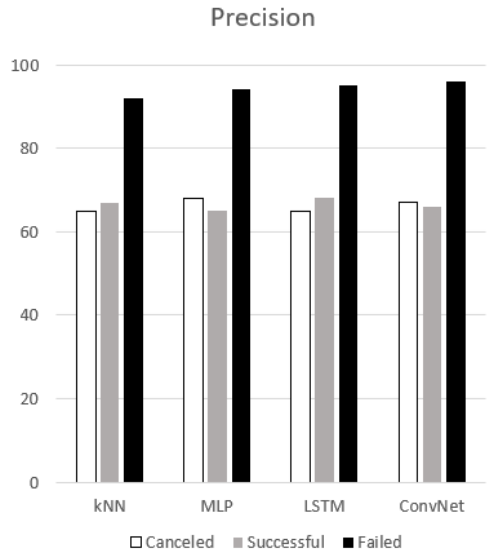

Figure 4: The precision in percentage for each class and each algorithm

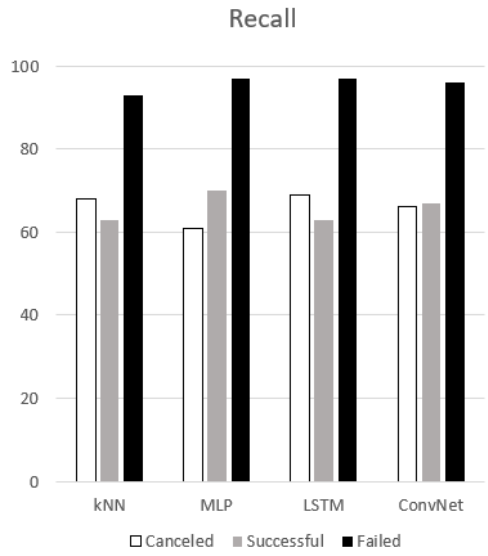

Figure 5: The Recall in percentage for each class and each algorithm

Another outstanding result are the bars of kNN and LSTM in Figure 4 and 5. We notice that these bars are almost exactly the same, but the predictions of LSTM are better.

Finally, we have a clear overview of the accuracy of the algorithms. As mentioned before we could not only base our conclusions on accuracy since we deal with an imbalanced dataset. Nevertheless, we can still take the results of accuracy into a count for a conclusion. Accuracy is the proportion of correct results that a classifier achieved, see Figure 6.

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + False\ Negative + True\ Negative}$$

Figure 6: The equation of accuracy

In Figure 7 we see an overview of all the accuracies for each algorithm. The kNN is doing a good prediction on the train ing set, but scores below on the test set. On average did the ConvNet predict the best of all the deep learning models, but LSTM predicts the best on unseen data. The LSTM model even increases in accuracy when it sees new data, it improves itself. Meanwhile, the ConvNet decreases in accuracy when seeing new data. The MLP the best prediction for precision and recall.

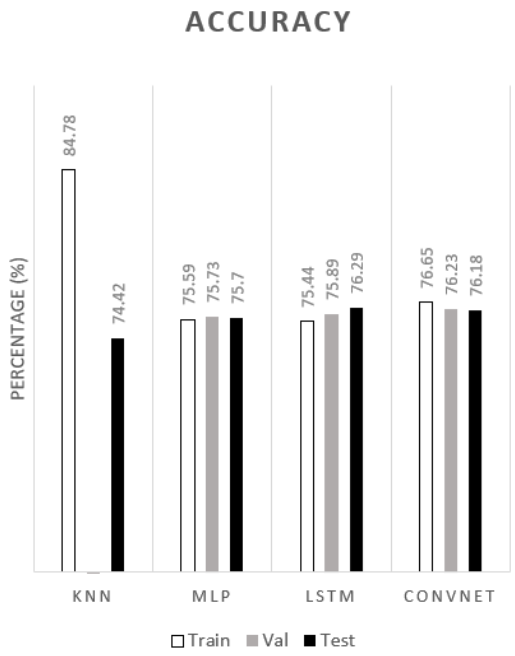

Figure 7: The accuracy in percentage for each set and algorithm.

5. General discussion

It is important to mention that ConvNets are normally used for predicting non-sequence or non-time series datasets. ConvNets exploit spatial relations, which means your features need to have spatial meaning. The order of the columns is important and cannot be changed. We would expect that this neural network will not predict very well regarding the fact that it does not fulfill the requirements.

This also counts for the LSTM model which received the most success when working with sequences. In this dataset is no sequence directly visible, but it is still possible to implement it an LSTM. Karpathy [2015] stated that: “The takeaway is that even if your data is not in form of sequences, you can still formulate and train powerful models that

learn to process it sequentially. You 're learning stateful programs that process your fixed-sized data." This is the reason why we can describe the success of a campaign as a sequence classification task. Due to this assumption, we are still able to use the deep neural networks but regarding the literature, it would not predict well.

The experimental results show that the prediction of LSTM and ConvNets are not as bad as we expect regarding the literature. They perform even better than an MLP and the kNN. If we can assume that we are dealing with a sequence classification task in this research, we state that the ConvNet is the best predictor overall. The best predictor according to the accuracy will be the LSTM, because it improves itself after seeing more new data. When we are not able to assume that we deal with a sequence classification task, we only can compare the results of kNN with those of the MLP. This classifier performs best according to recall and precision.

Most recent studies used machine learning classifiers to predict the success of a campaign. Several of these researches used a baseline accuracy of ± 68%. Since there are no similar studies done with the same dataset in the literature, we will also set 68% as a baseline for this research. We do have to find a similar experiment with the same dataset on GitHub[6]. The accuracy of the train data of this project for the MLP was 85.29% and they did not predict it on test data. Besides that, they did not take into account the imbalance of the dataset. This is a strength of this research in comparison to the experiment on GitHub.

Regarding the data, there is something suspicious about the classification of the class 'failure' as mentioned before. There could be two possible explanations for. First, there is a clear constancy in the data in order to classify this class. This would suggest that the classes for 'canceled' and 'successful' are just too difficult to predict with the seven features and 'successful' not. Second, the technique for improving the imbalance of the dataset lacks. In this research, we made use of a SMOTE method. This improved the classification compared to data without any method used, but the power of the major class, in the original data, is still too high.

6. Conclusion and further work

In this research, we studied the difference in the performance of deep neural networks compared to a machine learning technique. We did this by making use of a Kickstarter dataset and determined if we could predict the outcome of the project at the early stages.

In order to compare several deep learning models, we made use of algorithms that according to the literature could not be applied on this dataset, since it is not a data set with time series or sequences. According to the findings of this study, we suggest that it is possible to apply a ConvNet or LSTM on datasets which do not fulfill these requirements.

In this investigation, the aim was to assess if we can predict the outcome of a kickstart project better with deep neural networks then machine learning techniques. The difference between these the deep neural networks and the kNN are small. The MLP performs the best for precision and recall. At the same time does the result of LSTM and kNN look very similar for precision and recall but performs LSTM for each class in a small extent better. Ending with the ConvNet which has the best performance of all, because of its small difference between the classes, except 'failure', and its highest average accuracy.

In short, we can conclude that the deep neural networks can predict the outcome of a kickstart project better with deep neural networks than with machine learning techniques, but in a small extent.

We are encouraged about this result, but there are several future directions we would like to pursue. First, we should revise the dataset in order to find a better approach to improve the imbalance. Second, the hyperparameters of the optimizers for CNN and LSTM are not examined. Third, the suggestion that we can apply datasets that do not fulfill the requirements of a deep neural network should be tested on other datasets. Fourth, because this research is a deep learning experiment, we only focused on one machine learning technique. In order to avoid overgeneralization, we should compare the results of several machine learning techniques instead of one.

7. References

Greenberg, M. D., Pardo, B., Hariharan, K., & Gerber, E. (2013, April). Crowdfunding support tools: predicting success & failure. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems* (pp. 1815-1820). ACM.

Sawhney, K., Tran, C., & Tuason, R. (2016). Using Language to Predict Kickstarter Success.

Etter, V., Grossglauser, M., & Thiran, P. (2013, October). Launch hard or go home!: predicting the success of kickstarter campaigns. In *Proceedings of the first ACM conference on Online social networks* (pp. 177-182). ACM.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929-1958.

[6] A platform that brings together the world's largest community of developers to discover, share, and build better software; https://github.com/nalexus/Neural-Net-and-kNN----Kickstarter-projects-analysis/blob/master/Exploration%20of%20Kickstarter%20data.ipynb

Graham, B. (2014). Fractional max-pooling. *arXiv preprint arXiv:1412.6071*.

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Mollick, E. (2014). The dynamics of crowdfunding: An exploratory study. *Journal of business venturing*, *29*(1), 1-16.

Ackermann, N. (2018) *Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences*.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.

Karpathy, A. (2015). The unreasonable effectiveness of recurrent neural networks. *Andrej Karpathy blog*, *21*.

Chen, K., Jones, B., Kim, I., & Schlamp, B. (2013). *Kick-Predict: predicting Kickstarter success*. Technical report, California Institute of Technology.