Using Language to Predict Kickstarter Success

Kartik Sawhney **Caelin Tran** **Ramon Tuason**

Stanford University
Department of Computer Science
{kartiks2, cktt13, rtuason}@stanford.edu

Abstract

Kickstarter is a popular crowdfunding platform that is used by people to seek support on a variety of campaigns. Existing literature has already tackled the problem of predicting campaign success (meeting the funding target by the deadline), typically analyzing the evolution of a campaign's funding and number of backers over time to predict its success. While time-dependent features are effective in predicting success, especially as a campaign nears its deadline, we took a different approach to the problem and built a binary classifier that predicts the success of a campaign by analyzing campaign content, linguistic features, and meta-information at the time of its inception (through techniques such as sentiment analysis and latent Dirichlet allocation). With sufficient training, our classifier achieves over 70% accuracy on large test data sets. We show that language and campaign properties alone can be used to make moderate predictions about the success of a campaign, which helps us understand how the subtleties of persuasion and the choice of campaign topics can influence campaign donors and informs the process of evaluating and developing effective campaign pitches.

1 Introduction

Kickstarter is an internet service for people to raise money from crowdfunding. For a person to receive any resources, his or her campaign must meet its target by a deadline set at the time of publishing. In order to be successful (i.e., achieve full funding by the deadline), a campaign must effectively convey its mission to potential backers and persuade them to support it over other campaigns. Previous literature on predicting the success of Kickstarter campaigns has focused on applying machine learning to time series data (e.g., the amount of funds raised over time) and social networking. In this paper, we investigate whether it is possible to use information present at the beginning of a campaign to predict its success without using time-dependent data or data external to the Kickstarter campaign itself. We have built a binary predictor of Kickstarter success that focuses on actual content of a campaign and not its performance over time, capturing (1) people's initial reactions to the language used in a campaign and the description of its purpose and (2) the characteristics and types of campaigns that attract people.

2 Related Works

Literature on this topic can be broadly divided into three categories: specific work aimed at understanding crowdfunding dynamics, natural language understanding systems to predict audience reaction, and persuasion theories in general which can be used to develop our work further.

There have been several studies that leverage machine learning techniques to predict the success of a campaign. Vincent Etter et al. (see references section) analyzed the social network by constructing a projects-backers graph and monitoring Twitter for tweets that mention the project. They also discuss predictions based on the time series of early funding obtained. Combining features with social information helped to improve the model substantially. A study by Ethan Mollick on Kickstarter dynamics found that higher funding goals and longer project duration lead to lower chances of success, while inclusion of a video in a project pitch and frequent updates on the campaign increase the likelihood of full funding.

While these studies rely on time-dependent features and social information for predictions, recent advances in natural language understanding have helped make predictions based on linguistic considerations as well. For instance, studying the linguistic aspects of politeness, Danescu-Niculescu-Mizil et al. show that Wikipedia editors are more likely to be promoted to a higher status when they are polite. Furthermore, Althoff et al. analyze social features in literature to determine relations that can predict whether an altruistic request will be accepted by a donor. In addition, by analyzing the language and words used in daily conversation, tools such as LIWC (Pennebaker et al.) can be used for inferring psychologically meaningful styles and social behavior from unstructured text. Other resources like the NTU Sentiment Dictionary and SentiWordNet also help in opinion mining on text-based content (Pang and Lee).

Finally, we turn to more general theories of influence, noting the important role of reciprocity in social dynamics in investment situations as shown by Berg et al. In fact, Cialdini's iconic work on the psychology of persuasion demonstrates the significant influence of reciprocity in the form of limited rewards on the success of a Kickstarter campaign.

3 Implementation of our Predictor

3.1 Input-Output Behavior

The input-output behavior of campaign prediction is straightforward: after training a predictor using labeled (success / failure) Kickstarter campaigns, we use the predictor on arbitrary campaigns (input) and see if we expect them to succeed (output). For our raw dataset, we used Vitulskis et al.'s pre-existing, regularly updated, and labeled dataset created by crawling all Kickstarter pages.

The evaluation metric we used for a predictor is its classification accuracy of test campaigns. Given an input campaign, a predictor will classify it as a success (1) or failure (0).

As a concrete example, the following is a sample object representing a Kickstarter campaign that contains the relevant fields that our predictor will use. This campaign failed — perhaps its unoriginal idea and unexciting language were contributing factors.

```
{"name" : "Spinning Pots: Transformations of Clay through Human Hands",
"blurb" : "Creating a small business making functional pots/wares with clay and heat. Raising funds to purchase a kiln.",
"goal" : 3000,
"created_at" : 1431738817,
"deadline" : 1434680329}
```

Figure 1, Campaign Example: Note that the creation time and deadline are expressed as unix timestamps.

3.2 Baseline and Oracle

To test the reliability of lingual features in predicting a campaign's success, we implemented as our baseline a naive Bayes (NB) classifier that uses unigram occurrences as features. Unigrams are individual words, such as "spinning," "pots," and "transformations" from the sample above. We chose unigram occurrences as the features of our baseline since they provide a very basic analysis of language and content. They do not provide insight into how words are used or how they relate to one another. We randomly distributed campaigns in our dataset (160,000+ campaigns) into 80% training data and 20% testing data. Applying our classifier, we obtained a classification accuracy of 65%. This suggests that it is possible to classify Kickstarter campaigns with significant accuracy.

To determine an upper limit for the Kickstarter prediction problem, we implemented as our oracle a support vector machine that considers unigram occurrences along with information such as how many backers a campaign had by the end of its run, if it was featured in the Kickstarter website's "Spotlight" page, and if it was presented as a "Staff Recommendation." Applying our classifier, we got an average classification accuracy of 92%. The oracle illustrates the upper limit of any Kickstarter predictor based on data that it can receive from its start time to end time. This reflects our challenge of predicting a campaign's outcome just from information available at its inception. After all, if you incorporate more information as the campaign progresses, your prediction accuracy quickly increases, which is reflected in predictor performance in current literature.

We have a gap of about 27% between our oracle, which looks at the future, and baseline, which looks at language from the beginning of campaigns. However, the problem is not so intractable that language has no part in determining a campaign's success. Despite its simple nature, our naive Bayesian classifier actually performed above our expectations, yielding an accuracy well above the random chance threshold of 50%. Naive Bayes assumes that, given a certain success state ("True" or "False"), the unigram features of a campaign are conditionally independent. Of course, this assumption is not accurate because a campaign about a given topic is likely to have many related words and linguistic structures. This suggests that, with a more intelligent feature extractor and a more advanced predictor, we can further improve our classification accuracy.

4 Predictor Design and Implementation

In order to achieve better classification accuracy, we focused the design of our predictor on the need for more advanced features and a more advanced classifier. Our program consists of four stages: primary feature extraction, meta-feature assignment, training, and classification.

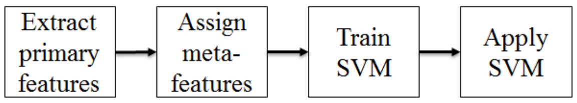

Figure 2, Program Flow: Our program has a straightforward linear flow that can be simplified even further into a training stage (primary feature extraction, meta-feature assignment, SVM training) and a testing stage (SVM classification). Our most novel contributions involve the feature extraction and meta-feature assignment components.

Our program interfaces with a JSON-formatted dataset through our primary feature extractor, which builds a list of campaign objects containing features of interest, such as unigrams, sentiment, Flesch-Kincaid readability score, and other language features. All of these features are campaign-specific, i.e., they may be extracted without comparison to any other campaigns. Some of the features are trivial to obtain, such as unigrams, but others require relatively complex natural language processing methods, such as sentiment analysis. The goal of this step in our program is to find features that tell us as much as possible about the topic of a campaign and the type of language it has.

The next step is the meta-feature assignment. Meta-features, or secondary features, come from algorithms that operate on the entire dataset to draw comparisons between campaigns, analyzing individual campaign features extracted in the first step. Examples of algorithms used here include latent Dirichlet allocation (LDA). Assignment consists of (1) applying algorithms to compare campaigns extracted from our training data and (2) assigning the results to those campaigns as features for use in predictor training.

Because there are tens of thousands of features that our data points can have, e.g., the presence of specific unigrams, most feature values that any given campaign has are equal to zero. Therefore, with our high-dimensional sparse datasets, we save ample memory with sparse matrices by storing only the non-zero components of each feature vector in memory. We use the sklearn library's FeatureHasher class, which uses a hash function to compute the matrix column corresponding to a specific feature name.

The last two steps are training and executing our classifier, a support vector machine (SVM). An SVM creates a trained hyperplane to divide test campaigns into successes and failures. We chose an SVM as the foundation of our predictor because of our need to process large numbers of features for each campaign and its ability to handle many different dimensions. This part of our program processes the features and labels of our campaign objects into sparse vectors, trains our SVM, and uses our SVM to predict the success of campaigns in our test dataset as well as the training dataset.

We use an SVM with a linear kernel instead of a non-linear one since we are essentially trying to solve a text classification problem in which we focus on the language used in the presentation of campaigns. Linear SVMs typically perform well with text classification because text is inherently discrete, sparse, and high-dimensional. With a very large number of features, SVMs generally do not see a notable improvement in performance when data is mapped to a higher dimensional space, which happens with SVMs that have non-linear kernels. In addition, given the massive amount of data that we process, we must also take speed into consideration, thus supporting the use of SVMs with linear kernels since they train relatively fast. We would also have fewer parameters to tune with grid search.

4.1 Tuning Hyperparameters

Hyperparameter optimization or model selection seeks to optimize a measure of an algorithm's performance on any given dataset by choosing a set of hyperparameters for a learning algorithm. This concept is distinct from actual learning problems, which instead optimize a loss function on a training set only. In other words, learning algorithms learn parameters that model or reconstruct their inputs well, while hyperparameter optimization seeks to ensure the model does not overfit its data by tuning certain values.

We performed hyperparameter optimization with grid search — also known as parameter sweeping — which simply searches through a manually-specified set of values within the hyperparameter space of a learning algorithm. Given that a grid search algorithm must be guided by some performance metric, we used internal cross validation on the training set.

Because we use an SVM classifier with a linear kernel, we tuned its regularization constant C. This parameter tells the SVM's optimization process how much we want to avoid misclassifying each training example. For larger values of C, the SVM will choose a smaller-margin hyperplane if that hyperplane leads to greater success with correctly classifying all the training points. On the other hand, smaller values of C will cause the optimizer to look for a larger-margin hyperplane, even if that hyperplane misclassifies more points. For very small values of C, getting misclassified examples is much more likely, even if the given training data is linearly separable.

To find an optimal value of C for our SVM given our entire training data set and full set of features, we searched through a list of logarithmically-spaced points between 10^{-6} and 10^{3}, ultimately getting an optimal value of approximately 0.007. Much smaller than the default value of 1.0, this value curtails overfitting and helps our classifier better accommodate randomized campaign test samples. This is useful given the innumerable ways language can be used.

4.2 Regularization

As SVM algorithms categorize multidimensional data and provide solutions that generalize to new data points, regularization algorithms seek to find the right balance between fitting training data and avoiding overfitting. Aside from the hyperparameter tuning done for C as discussed in the previous section, we also scale and translate each feature individually such that any given feature's value falls between -1 and 1, inclusive. This technique is useful because of the SVM's classification process. As it tries to minimize the distance between its separating hyperplane and its support vectors, certain features may overshadow others if they tend to have very high values that play a relatively large role in calculating the distance between their corresponding support vectors and the hyperplane. By looking at all the values for a given feature and scaling them to new values between -1 and 1 based on this distribution, we prevent any single feature from dominating a support vector's distance from the hyperplane.

5 Features

We chose the following primary and secondary features due to their contribution to classification accuracy and time limitations, which allowed us to achieve high performance with our SVM.

5.1 Unigrams

The first features we chose to use with our SVM were unigrams. To parse the title and summary of a campaign, we used the scikit-learn library's CountVectorizer class, which has a built-in tokenizer that parses strings into unigrams, assuring that different punctuation marks are handled correctly. Recall that unigrams were the only features used in our baseline, which achieved up to 65% accuracy, so we know that they are powerful features for predicting performance. We gain knowledge of a given campaigns subject and purpose based on the unigrams extracted from its title and summary as well as some understanding of its sentiment or mood. While unigrams gave us a rather high classification accuracy when used with naive Bayes alone, adding features helped provide more contextual information, such as fine-grained sentiment or language patterns, making this boundary even more concrete and well-defined.

5.2 Sentiment Analysis

We used scikit-learn's sentiment analysis algorithm, and trained it on a corpora of labelled movie reviews. By doing so, we can better understand the attitudes and emotions behind the messages of successful campaign authors. Rather than just using unigrams to determine sentiment, this algorithm uses deep learning techniques to perform sentiment analysis on whole sentences, helping us to capture the subtleties of sentiment more accurately. In addition, while we use sentiment value as a feature, we also use the probabilistic distribution to get a more comprehensive understanding and potentially resolve "neutral" sentiment.

5.3 Parts-of-Speech Tagging

To analyze any correlation between the success of a campaign and the sentence structure, we use the Stanford NLP group's POS tagger to parse parts-of-speech tags within a sentence. The generated POS tree helps us identify linguistic patterns within a sentence that lead to success. For instance, adverbs could be appealing, but too many adjectives and superlatives could detract from the efficacy of the campaign's message. This high-level structure, when combined with semantic understanding, helps provide the model with more comprehensive information for predictions.

5.4 Flesch-Kincaid Readability

Since Kickstarter campaigns should generally be accessible to a wide range of potential contributors, it becomes important to ensure that they are fairly easy to read, leading to our inference that the simplicity of the blurb positively influences the success of a given campaign. By using a combination of word length and sentence length as determinants of difficulty of the sentence, we apply the Flesch-Kincaid readability tests to the campaigns in our dataset, using the results as features. The results map to the approximate grade level of the audience that can understand the text. The readability score can be calculated as:

$$206.835 - 1.015\left(\frac{tWords}{tSent}\right) - 84.6\left(\frac{tSyll}{tWords}\right)$$

Here, 'tWords,' 'tSent', and 'tSyll' are short-hand for 'total words,' 'total sentences,' and 'total syllables,' respectively. A score between 100.00 to 90.00 corresponds to a 5th grade reading level, while a score between 30.0 to 0 corresponds to a reading level best fit for university graduates.

5.5 Latent Dirichlet Allocation

One of our secondary features came from implementing latent Dirichlet allocation, which is used to model text corpora and discover topics that are addressed by a document. The fundamental idea of the algorithm is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. These topics provide us with more semantic information than a simple term-frequency estimation or basic clustering technique. Furthermore, the fact that we identify multiple topics (as opposed to a single centroid in K-means, for instance), helps us extract more comprehensive information about a given campaign.

We divide our training dataset into two documents based on a campaign's failure or success. The hope is that the algorithm will be able to identify common topics that characterize these two classes of campaigns, boosting our predictions. For estimating model parameters, we use Collapsed Gibbs Sampling, and with a few optimizations such as the elimination of stop words, we get reasonably logical topics. For example, "creativity," "painting," and "art" constitute one topic. We then use the overlap of a campaign's title and summary with the words constituting the top ten identified topics as our features. Overall, LDA allows the SVM to better understand the different topics that a single campaign touches on.

5.6 Conditional Probability

One of our secondary features incorporates naive Bayes with unigrams, our baseline. We understood that we were not likely to achieve substantial performance gains by using this feature given that unigram occurrences were already being used as SVM features, but we wanted to combine the benefits of SVM as a more advanced classification algorithm with naive Bayes, which relies on conditional independence. While conditional independence is not completely true for our dataset, the relatively low degree of overlap (as shown by our naive Bayes performance) was convincing enough that we felt that we can potentially extract more information by including the magnitudes of conditional probabilities (probability of the input data given that we assume success, and the probability of the input data given that we assume failure) as features in our SVM.

5.7 Basic Campaign Data

While the preliminary stages of our project focus on the language and latent topics found in a campaigns title and summary, we also include some basic campaign data. Specifically, we include information on how much money a campaign is trying to raise and how much time it has between its inception and its deadline. After all, there are several key questions about the nature of crowdfunding that involve these pieces of information. For example, while it is reasonable to assume that smaller funding goals are easier to reach, there may be many relatively expensive campaigns whose missions better encourage the public to donate. Similarly, while campaigns with more time have more opportunities to be discovered by potential contributors, those with closer deadlines but worthwhile missions may give their backers a greater sense of urgency, encouraging them to reach out to others to help as well. All in all, these basic campaign data can hopefully give our model a better idea of trends that improve a campaign's chances for success.

6 Results and Discussion

On large datasets, we achieved around 71% test classification accuracy with our SVM once we fully loaded it with our features, beating the performance of our naive Bayes baseline (65%) and getting closer to our oracle's performance (92%).

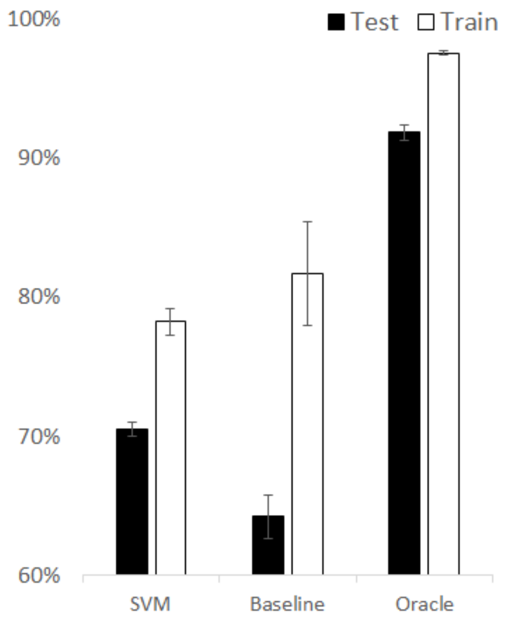

Figure 3, Overall Accuracy: This shows the average classification accuracy of the SVM, baseline, and oracle on the train and test datasets. Error bars represent standard deviations. The entire Kickstarter dataset (160,000+ campaigns) was used for three different randomizations of the train/test sets, which had an 80/20 split.

Note that the baseline actually had a better average training classification accuracy than the SVM, but this difference is not statistically significant ($p > 0.05$), and it might suggest that the SVM is better at generalizing the high volume of features from training data for application to test data.

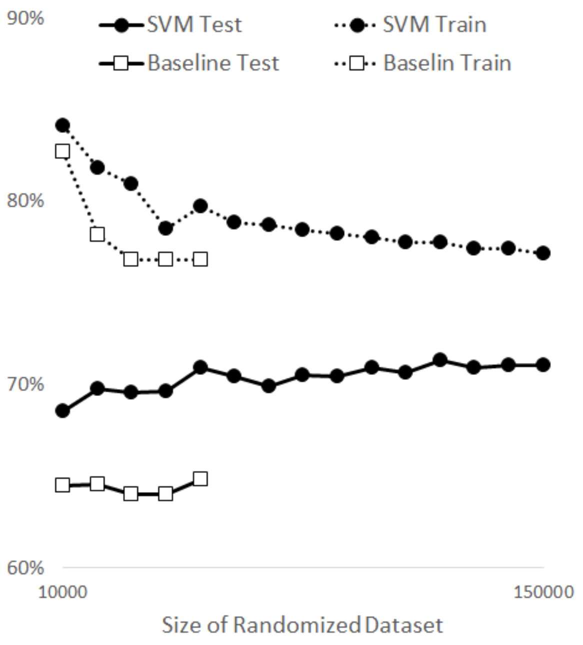

Figure 4, Accuracy vs. Dataset Size: This shows the average train and test accuracy of the SVM and baseline as we increase the size of the dataset (randomly chosen from the full Kickstarter dataset) in increments of 10,000. Due to time constraints, the baseline was only tested up to a size of 50,000 campaigns.

We see that the training accuracy of our SVM declines as the dataset size increases ($\rho = -0.86$), but the test accuracy increases ($\rho = 0.81$) until plateauing around 0.71%. This suggests that the generalization of the SVM improves as the volume of data grows.

We noticed that naive Bayes occasionally has better test accuracy on smaller datasets ($n \leq 10,000$, but it was rare for this to occur near 10,000), which is likely due to the SVM's need for ample amount of data to appropriately train its classification boundary. The accuracy of the baseline never gets far above 65%.

6.1 Time Performance

The time for our program to extract features, train, and test increased with dataset size in an $O(n^2)$

polynomial fashion. Analyzing meta-features was the most time-intensive component of our program — executing on the entire Kickstarter dataset took almost an entire day. The baseline was noticeably faster, especially with larger datasets. However, we do not believe that time performance is a very important metric in evaluating our predictor. This is because it is only necessary to update the model periodically once a significant quantity of new Kickstarter ventures complete their campaigns and, from the point of view of real-world application, it is only necessary to predict the success of a given campaign a single time: once it is created.

6.2 Performance by Category

Users must choose a category for their campaign at the time of publishing. Example categories include "Web," "Video games," and "Ceramics." We hoped that analyzing our dataset by category would provide insight into the performance of our predictor in specific fields. Unfortunately, no category had more than 10,000 campaigns, which is not enough for us to draw sound conclusions about trends in the performance of our predictor.

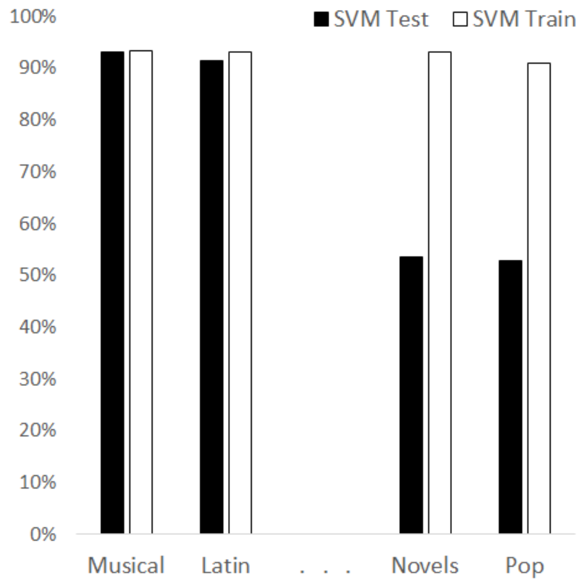

Figure 5, Accuracy of SVM on Individual Categories: For illustration of the range of performance on different categories, this shows the categories with the best and worst test classification accuracies.

On all categories, the test accuracy was 71% ± 11% with a range of 25% to 100%, and the training accuracy was 88% ± 4% with a range of 81% to 100%. The size of the dataset ranged from 11 to 4,001 campaigns with an average of 1,067 ± 1,148. On a category-by-category basis, there are not enough campaigns to adequately train our SVM, which shows in the wide range of results in test accuracy and the discrepancy with train accuracy. In general, the SVM performed better on larger categories, but this was also variable. For example, "Latin" is one of the smallest categories, but it had great test and train accuracies.

6.3 Discarded Features

In addition to the features that made it to the final version of our predictor, there were others that we considered and rejected due to complexity, issues with runtime performance, or lack of justification for increased predictor accuracy.

Regarding primary features, we considered including bigrams as well as unigrams. We thought that bigrams could provide a better indication of the kind of language that a campaign author can use to persuade others to contribute to his or her cause. However, bigrams added a significant number of features and did not contribute much more than unigrams, so we did not include them as features. We also omitted the author name feature, which was supposed to provide insight into authors with previously successful Kickstarters; location of the Kickstarter campaign author, which we realized just correlated with population density; and the length of the campaign description, which was subsumed by our Flesch-Kincaid readability feature.

We originally included K-means centroids as campaign meta-features. We obtained a number of topics (centroids) from K-means during the meta-assignment step of the training phase. During the testing phase, we assigned the closest centroid to each test campaign as a feature. The idea behind using K-means was to cluster algorithms based on textual similarity, hoping to identify interesting groups (corresponding to different campaign topics). However, because K-means assumes each campaign to have one "topic centroid," we instead use LDA, which allows each campaign to be represented as a combination of topics. Since we are still able to get important clustering information with added benefits of multiple topics, we decided to use LDA instead of K-means. K-means, as a meta-feature, also had a hugely detrimental effect on runtime performance before we removed it.

7 Future Work

7.1 Analysis of Video Transcript

As observed by Mollick, including a video of the proposed product or service in the Kickstarter campaign can increase the chances of it achieving full funding. Videos can also play a large role in drawing the audience emotionally, encouraging them to donate. Therefore, it would be interesting to run natural language parsing tools on video transcripts to evaluate them for pathos and overall effectiveness. More ambitious options would be to run computer vision techniques that classify or describe the objects being presented (e.g. beach, wholesome family, etc.), or to run audio analysis software that can give a good idea of what kind of sounds the video is presenting (e.g. soothing voice, action-movie background music, etc.).

7.2 Analysis of Campaign Mentions on Social Networks

Etter et al. analyzed the effect of social network interactions on campaigns by constructing a projects-backers graph and monitoring Twitter for tweets that mention campaigns. While our project currently relies on campaign characteristics at the time of its inception and does not monitor its popularity on social networks, it would be interesting to combine the two approaches, especially given the crucial influence of social networks. Other related factors may include the social media presence of a campaign author, or how popular its subject matter is in general.

7.3 Campaign Improvement Recommendations

Our SVM identifies the most important features that lead to a successful campaign, thereby classifying a test campaign as successful or not. The next logical step is to go beyond mere binary classification to logistical regression, scoring a campaign on various parameters. This in turn can be used to offer recommendations that help improve campaigns on certain well-defined aspects. For instance, one of the features could involve the most optimum use of certain adverbs in a sentence, and based on previous successful examples, the model can make suggestions to improve the campaign on this particular feature. Features can eventually be combined to help develop the most effective campaigns.

7.4 Role of Rewards

Cialdini et al. highlights the important role of rewards as an incentive for contributing to a campaign. It would be interesting to analyze this further by developing a strategy to associate value with rewards. Difficulties with this involve the different goals and priorities that different contributors can have, but it may be possible to extract trends regarding the nature of the rewards offered that can help with a campaign's success.

7.5 GloVe Representation

We would also like to make use of GloVe, an unsupervised learning algorithm created by the Stanford NLP Group for generating vector representations for words. Training for GloVe (Global Vectors for Word Representation) is performed on a given corpuss statistics on aggregated global word-word co-occurrence, and the representations that the model produces present insightful linear substructures of the word vector space. GloVe representation training models are effective and efficient tools for capturing co-occurrence information, given that they are decently fast to train and scalable to very large corpora. Even with small corpora and vectors with low numbers of dimensions, GloVe can still work reasonably well. For example, using GloVe reveals that the closest words to "frog" aside from "frogs" and "toad" are "litoria," "leptodactylidae," and "rana," which are all uncommonly-used words for different species of frogs. Hopefully, using GloVe will allow us to better understand the underlying semantic connotations found in titles and summaries.

7.6 Coreference Resolution

Coreference resolution refers to the task of clustering different instances of an entity being mentioned. This is particularly useful in NLP tasks, including retrieving information about specific named entities, and machine translation. In this context, we believe that this technique will help disambiguate pronominal references to previously mentioned entities, thereby helping our classifier better train with more specific mentions. This idea is inspired by "A Multi-Pass Sieve for Coreference Resolution" by Raghunathan et al.

8 Conclusion

By using metadata and linguistic features and fine-tuned classification algorithms, we were able to develop a binary classifier that can predict the success of a Kickstarter campaign using the information available at the time of creation with reasonable accuracy. This in turn can help improve the odds of campaigns achieving their goals. While we restrict ourselves to Kickstarter campaigns in this paper, this can broadly be understood as a text classification problem with multiple dependencies influencing the decision. Thus, our work can be used in various fields — from helping evaluate the effectiveness of slide decks for investors and VCs to providing general recommendations to improve documents. It will be interesting to investigate how the model changes with different applications and if there is a common framework that can be used for these related problems.

9 Acknowledgements

- Timothy Lee: Thank you for your suggestions and support of the project over the course of the quarter.
- Percy Liang: Thank you for helping us get the ground running with initial ideas and strategies.

10 References

Vincent Etter, Matthias Grossglauser, and Patrick Thiran. "Launch Hard or Go Home!" École Polytechnique Fédérale de Lausanne (EPFL). Lausanne, Switzerland.

Ethan Mollick. "The Dynamics of Crowdfunding: An Exploratory Study." The Wharton School of the University of Pennsylvania. Philadelphia, Pennsylvania.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. "A Computational Approach to Politeness with Application to Social Factors." Stanford University and Max Planck Institute SWS.

Tim Althoff, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. "How to Ask for a Favor: A Case Study on the Success of Altruistic Requests." Stanford University and Max Planck Institute SWS.

James W. Pennebaker and Martha E. Francis. *Linguistic Inquiry and Word Count*. Lawrence Erlbaum Associates, Inc. 1999.

Bo Pang and Lillian Lee. "Opinion Mining and Sentiment Analysis." Yahoo Research and Cornell University.

Joyce Berg, John Dickhaut, and Kevin McCabe. "Trust, Reciprocity, and Social History." University of Minnesota.

Robert Cialdini. *Influence: The Psychology of Persuasion*. HarperCollins Publishers, LLC.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers,Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. "A Multi-Pass Sieve for Coreference Resolution." Stanford University.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation." Stanford University.

Stanford Log-linear Part-Of-Speech Tagger. http://nlp.stanford.edu/software/tagger.shtml

Vitulskis et al. "Web Robots Kickstarter Datasets." https://webrobots.io/kickstarter-datasets/