Plead or Pitch? Predicting the Performance of Kickstarter Projects

Nihit Desai and **Raghav Gupta** and **Karen Truong**
Department of Computer Science, Stanford University
{nihit,rgupta93,truongk}@stanford.edu

Abstract

In this CS229 project, using 26K project proposals from crowdfunding platform Kickstarter, we evaluate the performance of different models (logistic regression, SVMs among others) on predicting whether a project will meet its funding goal or not, with a particular focus on features derived from the language used in project pitches. We then contrast the performance of our best model when using data from the projects' launch day against the last day of the funding period.

1 Introduction

The recent advent of web-based crowdfunding has connected project creators to backers around the world. While recent work has discussed patterns in factors correlating well with the success/failure of crowdfunding projects, not much has been made of formulating what constitutes a successful project pitch. While project metadata is important in determining project success, the language used in the pitch is also a major determinant of success as one of the few variables creators can tinker with pre-launch to maximize their chances of success.

Briefly, in this work, we explore the problem of forming attractive project pitches (focusing on the language used to describe the project and associated risks). The precise question we try to answer is: Given a snapshot of a Kickstarter project (note that projects span multiple days) as our input, consisting of text such as project description, risks and rewards as well as non-text metadata such as project category, creator bio etc, how accurately can we predict the final success of campaigns using different learning models (SVM, logistic regression, decision trees etc)? In addition, we also perform a qualitative analysis of the characteristics of language used in successful projects.

All three authors are crediting **part** of this project for CS224N: Natural Language Processing. In particular, **some parts of sections 3, 6 and 7**, and **very little from sections 4, 5** was included in our CS224N submission. Our central focus as regards the CS224N project was a rigorous explanation of linguistic techniques used in feature extraction, and qualitative analysis of the linguistic features' performance (which we touch upon very briefly in this submission). Section 4.3, which provides the mathematical explanation of our model choices, and our discussion of model performance, contained in Sections 7.3, 5.5 and 5.4 are **not** part of our CS224N submission.

2 Related Work

In this section, we give an overview of related work on crowdfunding and social analysis of text.

2.1 Crowdfunding dynamics

(Etter et al., 2013) analyze Kickstarter projects by constructing a projects/backers graph and extracting tweets mentioning the project, and present results on prediction based on the time series of early funding obtained. Focusing on Kickstarter dynamics, (Mollick, 2014) finds higher funding goals and longer project duration lead to lower chances of success, while including a video and frequent campaign updates increase the chances.

While the above work presents statistics about the determinant features for success, there has also been work on analyzing the language in project descriptions (Mitra and Gilbert, 2014). However, this only looks at projects when the campaign is no longer live. Here, we instead examine the first day of a campaign when the project creator still has the ability to make changes that increase their likelihood of success.

2.2 Text classification and analysis for social behavior

Text classification as a general problem has existed in various domains, and a variety of machine learning algorithms, including the ones we tried, have performed relatively well (Aggarwal and Zhai, 2015). Additionally, in recent years, scientists have been investigating the relationship between social behavior and language in online settings. For instance, studying linguistic aspects of politeness, (Danescu-Niculescu-Mizil et al., 2013) show that polite Wikipedia editors are more likely to be promoted. In another work, (Althoff et al., 2014) analyze social features in online communications to determine relations predicting whether an altruistic request will be accepted.

3 Domain and Dataset

We describe here the Kickstarter portal and funding model, followed by a description of our dataset.

3.1 Kickstarter

Kickstarter is an online crowdsourcing portal[1], with projects in diverse domains like technology, film and fashion. Kickstarter's all-or-nothing funding model makes it suitable for a prediction task. A typical project page has many parts relevant for our

[1] http://www.forbes.com/sites/chancebarnett/2013/05/08/top-10-crowdfunding-sites-for-fundraising/

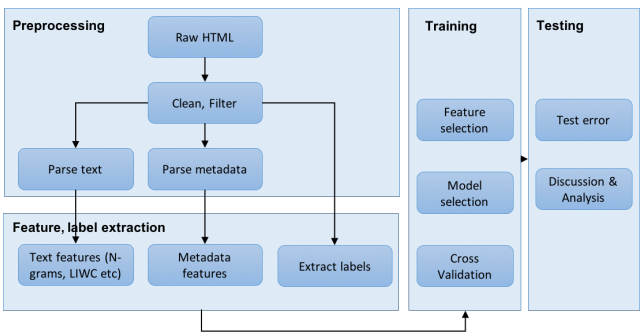

Figure 1: Data processing workflow

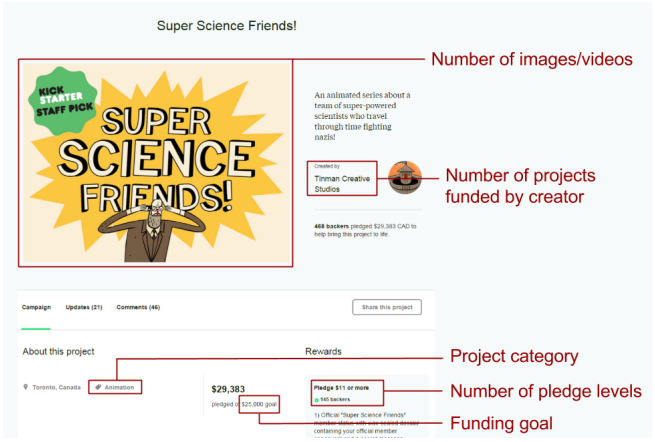

Figure 2: A sample Kickstarter project page showing metadata

task- project description (can contain text, images and video), potential risks, goal amount, pledge levels and user comments.

3.2 Data

We use a dataset of 26,543 Kickstarter campaigns whose final outcome (success or failure) is known. We obtained this data from Rob Voigt (Linguistics Dept, Stanford). For each project, the dataset comprises one snapshot per day of the project's Kickstarter webpage (HTML), while it's live. The crawl, which was run daily for a period of about a year starting June 2014, contains a daily snapshot of all projects active on that day. Note that for our precise problem definition, we are interested in the project pages' snapshots on the first and last days of the project's funding period. After cleaning, we parse the HTML, amounting to 80GB of raw HTML (using BeautifulSoup[2]) to extract four types of information: textual description, metadata (including but not limited to the title, start and end date, funding goal, image & video URLs), pledge information and the class label. The overall processing flow is outlined in Figure 1, and some important statistics about the data are presented in Table 1.

4 Experimental Setup

We now detail our experimental setup - problem definition, features and models. We model our task as a binary classification task, treating each project as a training (or testing) example, and with labels being *"success"* and *"fail"*, depending on whether the project met its funding goal or fell short.

4.1 Features

In this section, we describe our set of features. Results from feature selection experiments and a brief analysis of usefulness of various features is given in later sections.

4.1.1 Metadata

This feature set includes includes: project category, no. of videos, no. of images, no. of comments, no. of projects previously created/backed by creator and the no. of pledge levels and goal amount. We illustrate some of these in Figure 2.

4.1.2 N-grams

These are TF-IDF matrices of n-grams (n=1,2,3) from the project description and risks. We filter out phrases very specific to particular categories (e.g. 'arduino' for technology) to avoid feature correlation with project category feature. We tried using the plain counts matrix of the n-grams, but that led to a drop in performance.

[2]http://www.crummy.com/software/BeautifulSoup/

Metric	Successful	Failed
# Campaigns	7862	18681
Goal (avg)	$5747	$19344
# Sentences (avg)	24.5	23.6
# Reward types (avg)	8.73	7.52

Table 1: Kickstarter corpus statistics

4.1.3 Psycholinguistic features

We use LIWC: Linguistic Inquiry and Word Count (Pennebaker et al., 2001) to extract psycholinguistic features. LIWC is a lexical database of words relating to different psycholinguistic categories, examples being (*s denote stems)-

- Certainty: invariab*, factual*, absolutely
- Leisure: horseback, dvd*, celebrat*
- Achievement: effect*, persever*, founded

Our features are word counts in each category from description.

4.1.4 Sentiment from user comments

We use Stanford CoreNLP's sentiment analyzer (Manning et al., 2014) to annotate comments on the project webpage. The number of sentences with different sentiment scores ({0,1,2,3,4}, 4 most positive, 0 most negative) form our feature values.

4.2 Feature transformation

Many elements used in the objective function of our estimators (such as the RBF kernel of SVMs or the l_2 regularizers) assume that all features are centered around zero and have variance in the same order. Before training these models, we transform our data so that features have zero mean, unit variance.

4.3 Models

4.3.1 k-Nearest Neighbors (kNN)

These methods find a predefined number of training samples closest in distance to the new point, predicting the label from these. In k-nearest neighbors, a query point is assigned the class which has most representatives within the nearest k neighbors of the point.

4.3.2 Multinomial Naive Bayes (NB)

Naive Bayes is a popular supervised learning algorithm in text classification. Bayesian classifiers use the *maximum a priori (MAP)* decision rule to assign a label $\hat{y}$ as follows:

$$\hat{y} = argmax_y \; P(y) * \Pi_{i=1}^{n} P(x_i|y) \quad (1)$$

With multinomial Naive Bayes, the data distribution is parametrized by vectors θ_y for each class y, where each θ_{yi}= P($x_i|y$) has a multinomial distribution. The maximum likelihood estimate for θ_y is basically relative frequency counting:

$$\hat{\theta}_{yi} = \frac{\sum_{x \in T} x_i}{\sum_{i=1}^{T} \sum_{x \in T} x_i} \quad (2)$$

where the numerator is the number of times feature *i* appears in a sample of class y in the training set T, and the denominator is the total count of all features for class y.

4.3.3 Decision Trees (DT)

A decision tree classifier is non-parametric supervised learning method that poses a series of learned questions about the features of the training examples. Each time time it receives an answer, a follow-up question is asked until a conclusion about the class label of the record is reached, equivalent to the root-to-leaf traversal of a tree with the next child decided by certain feature values, and the leaves indicating the class labels.

4.3.4 Stochastic Gradient Descent (SGD)

Stochastic gradient descent is an optimization algorithm for minimizing an objective function. The SGD classifier is a linear model that updates and computes the gradient of the parameters using only a single training example (x_i, y_i) at time:

$$\theta = \theta - \alpha \nabla J(\theta; x_i, y_i) \tag{3}$$

We experiment with a hinge-loss cost function

4.3.5 Support Vector Machines (SVM)

SVMs are a popular discriminative algorithm, which has previously been used for text classification problems. The SVM attempts to find the max-margin hyperplane (defined by w and b; inference on sample x is $y = sgn(K(w, x) + b)$; K defined below) separating the dataset based on class, in a high-dimension feature space. Finding this optimal margin reduces to solving this convex optimization problem (reducable to quadratic program)-

$$argmin_{w,b} \frac{1}{2}||w||^2 + C\Sigma_{i=1}^{m}\epsilon_i \tag{4}$$

subject to constraints

$$y^{(i)}(K(w, x^{(i)}) + b) \geq 1 - \epsilon_i, \epsilon_i \geq 0 \forall i \in \{1, 2, ..m\} \tag{5}$$

where ϵ_is are slack variables to account for non-separability of the data, and kernel $K(x, y)$ is the inner product (in the high-dimension feature space) of vectors x and y. We try out two kernels - Gaussian/RBF, and polynomial with degree 2:

$$K(x, y)_{RBF} = exp(\frac{-||x - y||^2}{2\sigma^2}) \tag{6}$$

$$K(x, y)_{poly} = (x^T y + c)^d \tag{7}$$

4.3.6 Logistic Regression (LR)

This model has the hypothesis parametrized by θ:

$$h_\theta(x) = sigmoid(\theta^T x) = \frac{1}{1 + exp(-\theta^T x)} \tag{8}$$

Value of $h_\theta(x)$ is interpreted as probability of class 1 for example x. Inference is by $y = 1_{\{h_\theta(x) \geq 0.5\}}$. For training, we try logistic regression with L2 penalty ($C = 1$ being the L2 regularization hyperparameter):

$$\theta = argmin_\theta \frac{1}{2}\Sigma_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + C||\theta||_2^2 \tag{9}$$

5 Experiments

5.1 Procedure outline and evaluation

In experiments, we divide our data into training and test sets (90% and 10% respectively). We then use the training set to perform feature selection, model selection and parameter tuning. All experiments we performed using 5-fold cross validation on the training set. Finally, we report the performance of our best model on the test set. Our implementation for feature extraction, training and testing primarily uses scikit-learn (Pedregosa et al., 2011), a popular ML library in Python.

The metrics we focused on in our results was F-1 score, with added attention to the *successful* class' precision. For notational convenience, we use **P** for precision, **R** for recall, **F1** for F-1 score, **TP** for true positive, **TN** for true negative, **FP** for false positive, **FN** for false negative and the subscripts *s* for label=*successful*, *f* for label=*failed* and *o* for overall.

$$P = \frac{TP}{TP+FP} \quad R = \frac{TP}{TP+FN} \quad F1 = 2\frac{P*R}{P+R}$$

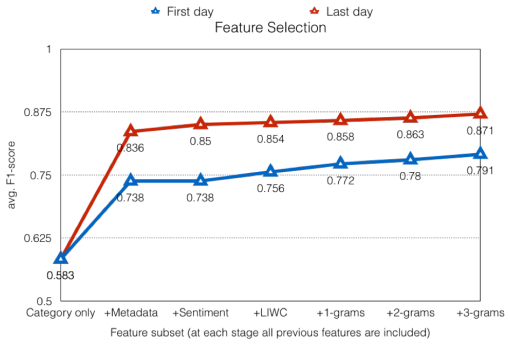

Figure 3: Performance of different feature sets with l_2-logistic regression with balanced class weights.

5.2 Feature selection

We do a **Forward Search** for feature selection (with the small change that we iterate over classes of features and not individual features). Starting with a simple model based only on the project category feature, we progressively add more features types and evaluate the performance of these features sets using the average F1 score of 5-fold cross validation. We use l_2-regularized logistic regression models with balanced class weights (for details see next section) for this exercise. Our choice of model for this experiment is driven by the popularity of logistic regression for text classification in general, plus its small running time.

Figure 3 shows the F1 increase as features are added. We perform feature selection experiments on project snapshots on the first and last day, owing to the fact that most predictive features at the campaign's start might be different from those more predictive as the campaign progresses. We observe that metadata is more important for predicting success at the end of a campaign compared to the start. Additionally, we find that user comment sentiment is only helpful for predicting success on the last day, which is expected since there are hardly any comments on the first day.

5.3 Model selection

	$\mathbf{P}_s$	$\mathbf{R}_s$	$\mathbf{P}_f$	$\mathbf{R}_f$	$\mathbf{P}_o$	$\mathbf{R}_o$	$\mathbf{F1}_o$
kNN	.34	.45	.82	.75	.71	.68	.69
NB	.57	.52	.78	.81	.71	.72	.71
SGD	.71	.48	.68	.85	.70	.69	.69
DT	.51	.58	.84	.80	.75	.74	.75
LR	**.75**	.62	.81	**.89**	**.79**	.79	**.79**
SVM_{poly}	53.	59.	84.	.81	.76	.75	.75
SVM_{rbf}	.71	**.65**	**.84**	.87	**.79**	**.80**	**.79**

Table 2: Performance of various models with the full feature set

Next, we train various models using the entire feature set and evaluate the performance of these models on snapshots on the first day of the campaign, using 5-fold cross validation. Something to note is that we balance class weights in our estimators since our dataset is skewed towards unsuccessful campaigns (after balancing, a "most common label" baseline would have an accuracy of $\sim$ **50%**).

Results (avgd over 5 fold cross validation) comparing the performance of various models are shown in Table2. To denote different models we use **kNN** for k-Nearest Neighbor, **NB** for Multinomial Naive Bayes, **DT** for Decision Tree, **SGD** for

linear classifier trained using SGD and hinge loss function, **LR** for l_2 regularized Logistic regression, $\mathbf{SVM}_{poly}$ for SVM with polynomial kernel and $\mathbf{SVM}_{rbf}$ for SVM with RBF (Gaussian) kernel. In addition, to the overall F1 score, we also consider the *successful* class precision (P_s) of the models in our evaluation. This is because if the end goal is to help project creators improve their project pitches, then we would prefer our model to have a reasonable P_s, ensuring that our model recommends useful changes to project pitches. We find that Logistic Regression (with l_2 regularization) and SVM (with Gaussian kernels) give the best results during cross validation.

5.4 Parameter tuning

We come to the problem of selecting parameters for our best performing class of models:

- We performed 5-fold cross validation on the training set for selecting regularization parameter C for SVM (because we observed disparity between the training and validation error); results are shown in Table 3. We observe that very small values of C seek only to maximize the margin, allowing for more examples to be misclassified. For $C = 1$ and larger, overall F-1 score remains fairly constant. However for large values of C, precision of *successful* label decreases and that of *failed* label increases- model is forced to make less training errors (overfitting). Since we care about *successful* class precision (P_s) of the models, which is why we report our final numbers on the test set for $C = 1$.
- For logistic regression, we observed that training and validation errors were very close, so we just used the default parameter value $C = 1$ (for l_2 regularization).
- For n-gram features, we selected the top-600 ngrams in an unsupervised fashion (using SVD on the TF-IDF matrix) to help speed up training of our models (especially SVM).

	$\mathbf{P}_s$	$\mathbf{R}_s$	$\mathbf{P}_f$	$\mathbf{R}_f$	$\mathbf{P}_o$	$\mathbf{R}_o$	$\mathbf{F1}_o$
C=0.1	.71	.57	.77	.86	.75	.75	.75
C=0.5	.73	.61	.80	.87	.77	.78	.77
C=1	.71	.65	.84	.87	.79	.80	.79
C=5	.62	.70	.88	.84	.80	.80	.80
C=10	.61	.70	.89	.84	.81	.80	.80

Table 3: Predictive performance of SVM (with RBF kernel) for varying C (regularization) parameter

5.5 Results on test set

5.5.1 First day and last day

Having experimented with feature sets and different models, we use these results to see how well can we predict success of a campaign as it progresses, using our two best models, SVM_{RBF} and Logistic Regression (LR). In Tables 4 and 5, we report the performance of our system on campaign snapshots on the first day (as a lower bound) and on campaign snapshots on the last day (as an upper bound). This is our final experiment for the prediction task and we use the test set which has thus far not been touched. For notational convenience, we use abbreviations as explained in Section 5.1. We observe that Logistic Regression performs better on the test set compared to SVM_{RBF} likely as a result of SVM_{RBF} overfitting (see next section).

5.5.2 Overfitting

To see if SVM_{RBF} and LR are overfitting, we evaluated their error on the training set, comparing with the test set error as shown in Tables 4 and 5. We observe that the model performance for either the first day or the last day data is very comparable on the training and test set, for LR. However, for SVM_{RBF}, the training error is much less than the test error for both the first day and the last day. This leads us to conclude that SVM_{RBF} is overfitting the data while LR isn't. This is despite our efforts to prevent it (we tuned parameter C, and also tried setting a bound on number of support vectors).

	$\mathbf{P}_s$	$\mathbf{R}_s$	$\mathbf{P}_f$	$\mathbf{R}_f$	$\mathbf{P}_o$	$\mathbf{R}_o$	$\mathbf{F1}_o$
Test_{First}	.72	.63	.84	.87	.79	.79	.79
Test_{Last}	.89	.74	.87	.95	.88	.87	.87
Train_{First}	.95	.85	.93	.98	.94	.93	.93
Train_{Last}	.97	.86	.93	.99	.95	.95	.94

Table 4: SVM_{RBF} performance on test set

	$\mathbf{P}_s$	$\mathbf{R}_s$	$\mathbf{P}_f$	$\mathbf{R}_f$	$\mathbf{P}_o$	$\mathbf{R}_o$	$\mathbf{F1}_o$
Test_{First}	.76	.63	.81	.89	.79	.79	.79
Test_{Last}	.89	.76	.88	.95	.88	.88	.88
Train_{First}	.79	.65	.82	.90	.81	.81	.81
Train_{Last}	.89	.76	.88	.95	.89	.89	.89

Table 5: Logistic Regression performance on test set

6 Challenges

Relying only on linguistic features for classification is tricky since all projects aspire to succeed, making the task hard even for humans, unlike a problem like spam classification, where the spam language is distinguishable from normal language.

For some campaigns, description is entirely through images. Since open source frameworks for extracting text from images (we tried Tesseract[3]) are not robust enough for our needs, projects falling in this category are often misclassified as unsuccessful (owing to the data's inherent skew).

During parsing we observed that some HTML pages did not confirm to the normal structure. As a result, some attributes from these pages weren't parsed correctly, leading to missing feature values.

7 Discussion

7.1 Linguistic features

Interestingly, we find that most of our highest positive and negative weighted features are in fact n-grams. In this section, we analyze linguistic features highly predictive of success of a Kickstarter campaign (refer to Table 6 for a more extensive list) and try to offer some insights in this regard:

- **Reciprocity**: Reciprocity is the tendency to return a favor in exchange for receiving one. We find many predictive phrases that are often used to offer a reward or a gift in return for donation funds.
- **Social relationship**: We find that phrases indicative of success employ social dynamics and provide context in which their projects, if successful, will have a positive impact.
- **Emotional appeal**: Successful campaigns are seemingly emotionally appealing to the readers, and both negative and positive emotion are well represented.
- **Gratitude**: Successful campaign text often conveys gratitude towards the backers.

[3]https://code.google.com/p/tesseract-ocr/

Group	Phrase list
Reciprocity	free shipping, you receive, early bird, be the first, your reward
Social	friends, friendship, community, and family, his family, people
Emotion	passion, dream, inspired to start, believe that, impact, volunteer
Thankful	thank you, so thankful, thanks, thanks so much, grateful, grateful for your
Pitch	why support, funds will cover, will be used, aiming to, aim to raise
Collective	help us, we can, we raise, we plan to, we need, we found, we created

Table 6: Phrases in project descriptions most predictive of successful Kickstart projects, grouped

- **Collective phrasing**: We find that project descriptions making use of the singular first-person pronoun *'I'* tend to belong to unsuccessful projects, while those using the plural pronoun *'we'* are generally successful.

Feature weights for LIWC categories corroborate these observations; categories representing emotion (*positive emotion*, *negative emotion*), social processes (categories *friend, family*) are strong positive predictors. At the same time, membership in LIWC categories like *death, sad, anger* is indicative of failure.

7.2 Metadata Features

When training and testing on the first-day project descriptions, only the feature on goal amount (with a high -ve weight) features among the top 50 features. Surprisingly, the features on the number of images and videos, while having a moderate positive weight, did not affect classification performance drastically. But when training and testing on the last-day data, our features on the number of backers and the number of comments gain prominence, and feature among the top 10 most predictive features.

7.3 Models

We discuss possible interpretations of the performance of various models on our dataset. Here, we'd like to point out that our prediction task is more naturally modeled using a discriminative approach rather than a generative one. This is because all campaigns aspire to succeed. Therefore, it is better to model conditional probability of the label given the data as opposed to their joint probability.

7.3.1 k-Nearest Neighbors

Given the dataset size, we expected kNN to provide a decent baseline for our task. However, with $P_s \sim 0.35$ and $F1_0 \sim 0.68$, it does not perform well. We reason this is because campaigns fairly similar in terms of the language may have different outcomes; only a handful succeed, biasing kNN towards predicting failure most of the times.

7.3.2 Multinomial Naive Bayes

Notwithstanding being a generative model where our problem is better modeled discriminatively, Multinomial Naive Bayes performs commendably on this task. However, it is a simple model with strong assumptions and often a high bias (as discussed in the lectures). It performs very consistently during cross-validation ($P_s \sim 0.55$, $F1_o \sim 0.7$), helped by the size of the data, and provides a competitive baseline for the task.

7.3.3 Decision Trees

Decision trees are not too useful in problems with high-dimensional feature spaces (e.g. text classification) where decisive features may not exist or be hard to find, due to the sparsity of the feature set (particularly the n-gram features in our case). In addition, two projects with fairly similar feature values may still have different labels/outcomes, which is a disadvantage for decision trees, hereby unable to distinguish such examples.

7.3.4 SGD Classifier

The SGD Classifier presented interesting results. During 5-fold cross validation, the F1 score for the *successful* class fluctuates wildly, from 0.1 to 0.6 and the overall F1 score from 0.66 to 0.81 without converging within 50 epochs (default value for *scikit-learn* is 5). We hypothesize this is because of SGD being unstable with non-separable data; a largely continuous presence of outliers during training would guide SGD towards fluctuating directions. This is supported by the linear kernel SVM not converging within 70k iterations (discussed in Section 7.3.5).

7.3.5 SVMs

SVMs provide good results here on our test set, only slightly inferior to LR on P_s. The radial basis function kernel outperforms the polynomial kernel significantly, while the linear kernel fails to converge in reasonable time (see Section 7.3.4). We performed 5-fold cross validation for selecting the parameter C (details in Section 5.4) and our reported results are with $C = 1$, which offers the best tradeoff. One limitation we observed in case of SVMs was the overfitting on training set.

7.3.6 Logistic Regression

One of the two most popular linear models (the other being SVMs) in the text classification domain, our logistic regression models provides good performance, with precision for the *successful* class- $P_s \sim 0.75$ and $F1_o \sim 0.79$ for snapshots on first day of the campaign. Our l_2-regularized logistic regression model comes up with feature weights that are explainable- it forms the basis for our feature search. Another advantage with logistic regression is that its output can be interpreted as probability of a campaign being successful.

8 Conclusion and Future Work

We present results on Kickstarter project success prediction using mainly the initial description, indicating a departure from prior work relying on post-launch data. We identify Logistic Regression and SVM_{RBF} as our top models, the former avoiding overfitting to boot, and also illustrate the efficacy of metadata and linguistic features for the task, obtaining good results on the initial description and identifying stylistic patterns in the language used in successful campaigns.

Apart from the feature set extensions discussed in Section 6, we discuss directions for future research in this domain. One interesting extension to this work would be to use the obtained feature weights themselves to suggest to project creators which portions of a project description are lacking. A thorough analysis of backers' preferences for different kinds of rewards, and its impact on project success, would be interesting to explore. Finally, we would like to get to the root of the SVM's overfitting problem, given more time.

References

Charu C. Aggarwal and ChengXiang Zhai 2012. *A survey of text classification algorithms*, Mining text data. Springer US. 163-222

Christopher Manning, Surdeanu Mihai, John Bauer, Jenny Finkel, Steven Bethard and David McClosky. 2014. *The Stanford CoreNLP Natural Language Processing Toolkit*. Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec and Christopher Potts. 2013. *A computational approach to politeness with application to social factors*. Proceedings of the Conference of the Association for Computational Linguistics, 2013.

Ethan Mollick. 2014. *The dynamics of crowdfunding: An exploratory study*. Journal of Business Venturing 29.1:1-16.

James W. Pennebaker, Martha E. Francis, and Roger J. Booth.. 2001. *Linguistic inquiry and word count: LIWC 2001*. Mahway: Lawrence Erlbaum Associates 71.

Joyce Berg, John Dickhaut and Kevin McCabe. 1995. *Trust, Reciprocity, and Social History*. Games and Economic Behavior: 10.1:122-142.

Tanushree Mitra and Eric Gilbert. 2014. *The Language that Gets People to Give: Phrases that Predict Success on Kickstarter*. Proceedings of ACM conference on Computer-Supported Cooperative Work and Social Computing.

Tim Althoff, Cristian Danescu-Niculescu-Mizil, Dan Jurafsky. 2014. *How to Ask for a Favor: A Case Study on the Success of Altruistic Requests*. Proceedings of the International Conference on Weblogs and Social Media.

Vincent Etter, Matthias Grossglauser, and Patrick Thiran 2013. *Launch hard or go home: predicting the success of Kickstarter campaigns*, Proceedings of the first ACM conference on Online social networks.

Fabian Pedregosa et al. 2011. *Scikit-learn: Machine learning in Python*, The Journal of Machine Learning Research (JLMR)