CS5228 Project Report

KDD Cup 2014 - Predicting Excitement at DonorsChoose.org

1. Introduction

In this report, we describe our effort and our solution to a problem hosted on Kaggle as part of KDD Cup 2014. Our solution is **ranked *7th best* on the Kaggle private leaderboard**. We describe our team and team member's contributions in Section 2. Section 3 gives an overview of the problem presented in the competition and its challenges. Section 4 and 5 describes the data and filtration. In Section 6, we present our solution to the problem. Our experiments and results are described in Section 7. Finally, Section 8 concludes the report.

Completed • $2,000 • 472 teams

KDD Cup 2014 - Predicting Excitement at DonorsChoose.org

Thu 15 May 2014 – Tue 15 Jul 2014 (3 months ago)

Dashboard

Private Leaderboard - KDD Cup 2014 - Predicting Excitement at DonorsChoose.org

This competition has completed. This leaderboard reflects the final standings.

See someone using multiple accounts? Let us know.

#	Δ1w	Team Name * in the money	Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑1	'STRAYA *	0.67814	213	Tue, 15 Jul 2014 00:21:34 (-0.2h)
2	↓1	DataRobot *	0.67320	220	Tue, 15 Jul 2014 23:32:50 (-2d)
3	↑30	ChaoticExperiments (KIRAN R) *	0.67297	69	Tue, 15 Jul 2014 19:35:05 (-2d)
4	↓1	dkay & bmax & James King	0.66473	239	Tue, 15 Jul 2014 23:26:11 (-2.1d)
5	↓1	Triskelion,Yan, KazAnova & Shize	0.65949	225	Tue, 15 Jul 2014 23:29:42 (-0.4h)
6	↑35	Giulio, orchid, Luca & Ben	0.65919	264	Tue, 15 Jul 2014 18:51:21 (-0.4h)
-		**Panda Delights**	**0.65557**	-	**Sun, 02 Nov 2014 10:19:52** Post-Deadline
Post-Deadline Entry If you would have submitted this entry during the competition, you would have been around here on the leaderboard.					
7	↑2	:-)	0.65372	123	Tue, 15 Jul 2014 22:41:25 (-4d)
8	↑8	柳景明	0.65367	53	Mon, 14 Jul 2014 09:08:13 (-41.1h)

2. Our Team

We are a team of three members. Our team is registered on Kaggle as '**Panda Delights**'. Raymond Hendy Susanto wrote the code to perform feature extraction and learning. Muthu Kumar preprocessed the raw text data, extracted unigram features and added part-of-speech information. Peter Phandi implemented a time-based discounting function as a post-processing step. We frequently had meetings throughout our project to discuss ideas, analysis and directions to do the project. Finally, all of us contributed in writing this report.

3. Problem Description

DonorsChoose.org is an online charity website to help students in need. The teachers can post students needs at their school as a project on the website. Donors can then see the description of the project which includes various quantitative data as well as an essay to choose which project(s) to fund or donate to according to their interest. After the project is approved for funding, the items that the project needs will be shipped to the students.

In this task, DonorsChoose.org wanted to know which proposed projects are considered "*exciting*" by the donors. That is, we need to train a supervised classifier C to predict if a new proposed project is exciting or not using the project's description and various other meta data. The task is a binary classification problem.

Knowledge of this prediction could help DonorsChoose.org automatically recommend projects for funding. Further to reducing the information overload on the donors to read a vast number of project proposals, this could improve the funding outcome and user experience due to the system's potentially non-arbitrary recommendations.

One of the main challenges we faced when dealing with this task was the large number of data fields provided in the dataset. This makes it difficult to know which data are relevant to the "excitingness" of a project.

4. Data

We are provided with 3 data files with training and test data: projects.csv, essays.csv, resources.csv and 2 more data files: donations.csv and outcomes.csv with only the training data. Since detailed descriptions of this data are provided on the Kaggle platform, we do not repeat them here.

To build our predictive models we only used data from projects.csv and essays.csv. We only used files that had both training and test data to avoid overfitting to the training data.

Training and test data are divided by time. All projects posted prior to 1st January, 2014 are provisioned as training data. The rest is to be used as the test set.

5. Data Filtration

From initial experimental results (described in Table 1) we find that most of the historical data in the training set overfits the model. Performance on "All" data in row 1 of Table 1 has the best performance on the training data but its score on the public leaderboard is the worst among various other datasets. Row 3 of Table 1 shows the best performance on the public leaderboard. This corresponds to data from projects posted from July 2012 until 1 January 2014. So we filtered the training set to include only these projects. All forthcoming results in this report used this training data.

Training set Start date	**Training Set Size (# projects)**	**3-fold CV Over Training Data**	**Kaggle public leaderboard score**	**Kaggle private leaderboard score**
All	619,326	0.76894	0.58110	0.63825
January 2012	248,955	0.67067	0.60727	0.65418
July 2012	196,770	0.65882	0.60818	0.65557
January 2013	131,329	0.64538	0.60417	0.63495

Table 1: Experiments to find out the best training data set

Post filtration the training and test data split are as follows:

# projects in the test set	44,772 (~ 19%)
# projects in the training set	196,770 (~ 81%)
Total	241,542

Table 2: Training and test split

Due to the nature of the task our training data is quite imbalanced. The number of negative instances (projects labelled '*not exciting*') far outnumbers the number of positive instances (projects labelled '*exciting*').

Positive instances (exciting)	18,670 (~ 9%)
Negative instances (not exciting)	178,100 (~ 91%)

Table 3: Number of positive and negative instances in the training set

6. Approach

In this section, we describe our approach to build a model to predict if a student project is *exciting* given its proposal and other project meta data such as details of the school proposing the project, resources requested and the like. We list out in the next section the features used in our model and the learning algorithms we used to train our model.

6.1. Features

Our features are derived from the project details and the proposal essay provided in the competition. We categorize the features in “projects.csv” file into three types: binary, numeric, and categorical features. We further included text features extracted from project essays in “essays.csv”.

Binary Features

A binary feature contains two possible values (i.e., “true” or “false”). We extracted the following binary features from those provided:

- school_charter - whether a public charter school or not (no private schools in the dataset)
- school_magnet - whether a public magnet school or not
- school_year_round - whether a public year round school or not
- school_nlns - whether a public nlns school or not
- school_kipp - whether a public kipp school or not
- school_charter_ready_promise - whether a public ready promise school or not
- teacher_teach_for_america - Teach for America or not
- teacher_ny_teaching_fellow - New York teaching fellow or not
- eligible_double_your_impact_match - project was eligible for a 50% off offer by a corporate partner (logo appears on a project, like Starbucks or Disney)

- eligible_almost_home_match - project was eligible for a $100 boost offer by a corporate partner

Numeric Features

A numeric feature takes an integer or real value. For example, the number of students that are potentially impacted by the project (students_reached) and project costs. We used the following numeric features in the data:

Cost of implementation
- fulfillment_labor_materials - cost of fulfillment
- total_price_excluding_optional_support - project cost excluding optional tip that donors give to DonorsChoose.org while funding a project
- total_price_including_optional_support - see above

Projected project impact
- students_reached - number of students impacted by a project (if funded)

Categorical Features

A categorical feature takes one out of a fixed number of possible values. For example, a school's poverty level (poverty_level) can take one out of four possible values (highest, high, moderate, or low). Most attributes in the data are categorical. We used the following categorical features:

Teacher information
- teacher_acctid - teacher's unique identifier (teacher that created a project)

School information
- school_latitude
- school_longitude
- school_state
- poverty_level - school's poverty level.

Project materials
- primary_focus_subject - main subject for which project materials are intended
- primary_focus_area - main subject area for which project materials are intended
- secondary_focus_subject - secondary subject
- secondary_focus_area - secondary subject area
- resource_type - main type of resources requested by a project
- grade_level - grade level for which project materials are intended

An important point to note when using categorical features is that most machine learning algorithms expect continuous output and will treat the categories as ordered, which is sometimes not desired. To avoid this invalid assumption, we preprocessed the categorical features to encode them using ***one-of-K or one-hot encoding scheme***. This scheme encodes each categorical feature with *m* possible values as *m* binary features, with only one feature active. This also results in a sparse feature representation.

We observe that some features such as the school location specified by the school's state have a large number of categories. Due to one-hot encoding each of these categories is mapped as a feature. We only used some of these features and omitted the rest that were not helpful in improving performance. The omitted features were too sparse to affect performance.

We treated school's latitude and longitude as categorical variables instead of numeric. Each latitude and longitude specifies a school's location uniquely. We do not bin the full range of latitudes and longitudes and assign categories to bins. We instead treat each school's latitude and longitude as a category by itself. We then encode it using one-hot encoding like other categorical features. We found that school's location information is important and treating them as categorical works better.

Bag-of-words features from text data

We used the essay text from the project proposals to extract unigram features and their part-of-speech (POS) tags. There were 196,770 essays in our post-filtration training corpus. We treat each individual essay as a document.

We segmented essays by sentences. Every sentence is further tokenized. We discarded non-English tokens and other punctuations. There were **68,409,639 (~ 68.4M)** tokens or word occurrences in total from **102,027 word types (~ 100K)**.

For each term thus extracted we compute their term frequencies (tf) per document.

Term frequency (tf): Number of times a term *t* occurs in a document *d*.

We also compute the corpus wide document frequency (df).

Document frequency (df): Number of documents in corpus *C* in which the term *t* occurs.

The inverse document frequency (idf) captures the importance of a term in the corpus. Since some terms occur more rarely than others, documents containing such terms are considered more important than the rest.

Inverse document frequency (IDF) of a term *t* = ln (# documents in the corpus / df)

We then compute *tf-idf* weights for each term and index them in a text database.

Term weight, tf-idf of a term *t*, = tf x idf.

Every document is represented, in the vector space, as a term vector specified by its term weights. This term vector captures the notion of document similarity and dissimilarity. That is, similarly worded documents have vectors in the same direction in the vector space.

The magnitude of the term vectors would be longer for long documents than short documents. Since we are only interested in the direction of the vector to identify documents that are not similar to the rest of the corpus we normalise for document length by turning the term vector into its unit vector. This is done by dividing each component of the term vector by L2-norm of term vector *T*.

The term vector representations are used to train a Max-Ent classifier to predict the probability of project being *exciting* given its essay text.

Next we also augment the term vector by POS tags of unigrams we extract from the essays. POS tags generalize over word types by grouping them into of the syntactic types of according to English grammar.

6.2. Learning Algorithms: Maximum Entropy (Max-Ent) model

Max-Ent is a discriminative classification algorithm that works on the principle of maximum entropy. Max-Ent gives state-of-the-art performance in many applications, such as part-of-speech tagging [1] and text classification tasks [2].

The Max-Ent principle [1] states that the probability distribution best fits the given data is the one with the maximum entropy.

Our model is a L2-regularized Max-Ent model. Given a set of training instances (x_i, y_i), it tries to learn a weight vector W that minimizes the following cost function:

$$\min_w \tfrac{1}{2} w^T w + C \sum_i \log(1 + e^{-y_i w^T x_i})$$

6.3. Ensemble Classifier

We trained two maximum entropy classifiers based on two different feature sets. The first classifier *C-project* was trained using boolean, numeric, and categorical features from the project.csv file. These are meta-data about the project. The second classifier

C-text was trained using bag of words from the project proposal essays and their POS tags.

To get the final prediction from both these classifiers, we simply average the two probability values from the two classifiers:

$$P_{final}(y = exciting|X) = \frac{P_{c-project}(y=exciting|X) + P_{c-text}(y=exciting|T)}{2}$$

6.4. Post-processing

The dataset on Kaggle was released on 15th May 2014 and teams were allowed make submissions thereafter. However, we noticed that some of the projects in test data were posted to the donors only by 12th May 2014. We hypothesize that these projects would not have had sufficient time to attract the attention of the donors to be judged as *exciting* on the same scale as the projects that had been submitted much earlier. They stand a higher chance of being unfairly judged as *not exciting* by DonorsChoose.org. This creates a bias in the test data.

To address this problem, we propose a linear time-discounting scheme for projects in the test data. We discount the excitement probability value of each project by how close they were to the data release date, 15th May 2014, of the competition. That is, given a classifier probability score for an exciting project $P_{clf}(y = exciting|X)$, we define the discounted probability $P_{discount}(y = exciting|X)$ as:

$$P_{discount}(y = exciting|X) = P_{clf}(y = exciting|X) * D$$

where

$$D = \alpha \times \frac{number\ of\ elapsed\ days}{total\ number\ of\ days} + (1 - \alpha)$$

We set $\alpha = 0.9$ and apply discounting for projects starting from 10 February 2014. We apply this discount to all results we present henceforth. We show the effects of discounting in contrast to undiscounted results using the best set of features in Table 6. It can be observed the discount improves performance that is significant to the naked eye.

6.5. Implementation

We implement our solution in Python 2.7. Our experiments were run in Ubuntu 14.04. We used the logistic regression implementation in scikit-learn (http://scikit-learn.org), namely sklearn.linear_model.LogisticRegression. We used NLTK library for text processing and part-of-speech tagging.

7. Experiments and Results

In this section, we describe our experimental settings and results. Each approach is evaluated using ***3-fold cross-validation*** on the training data. Furthermore, we also evaluate the performance on the Kaggle public and private leaderboards. Note that during the competition, only the public leaderboard score is made available, while the private leaderboard score was hidden. Thus, to ensure fairness to those existing projects on the leaderboard, our decisions are based solely on our cross-validation results and public leaderboard scores. To evaluate each system's performance, we compute the ***area under the curve (AUC)***, which is the official evaluation metric that was used in the competition.

To engineer features to produce the best performance, we experiment with various feature sets. We first built a classifier using boolean features only. Next we added numerical and categorical features. We then trained a separate classifier which uses bag-of-words features extracted from the project essay. The probabilities from the two classifiers are averaged, as explained in the previous section.

Table 4 shows experimental results with different feature sets. We can observe that adding more features improves the system performance on both 3-fold cross-validation and public leaderboard score. Note that we did not apply discounting for cross-validation, since we only use provided training data for cross validation.

Features	**3-fold CV Over Training Data**	**Kaggle Public Leaderboard Score**	**Kaggle Private Leaderboard Score**
boolean features	0.55554	0.58706	0.62673
+ numerical features	0.56524	0.58742	0.62393
+ categorical features	0.65185	0.60136	0.64924
+ essay	0.65856	0.60812	0.65551
+ pos	0.65882	0.60818	0.65557

Table 4: Experiments with different feature sets.

Term vectors are typically long and results in large number of features increasing the model complexity. Often these vectors are sparse since many of the terms do not occur in all the documents. Some terms are occur so infrequently that removing them from the feature set will make no difference. This type of feature selection is implemented by cutting-off terms that occur in less a minimum number of documents, min_df. To find the

best min_df parameter we experiment with in of increments starting from 1. Results are tabulated in Table 5. Clearly a frequency cut-off of up to 3 results in no performance drop in the public leaderboard score while the feature vector size is more than halved. This is the best Max-Ent given the term vector.

min_df	size of the term feature vector	3-fold CV over training data	Kaggle Public Leaderboard Score	Kaggle Private Leaderboard Score
1	71,484	0.65861	0.60788	0.65551
2	45,345	0.65858	0.60801	0.65551
3	37,168	0.65856	0.60812	0.65551
4	32,843	0.65858	0.60810	0.65552
5	29,980	0.65859	0.60812	0.65549

Table 5: Experiments with different document frequency cut-off (*min_df*).

Test Data Type	Kaggle Public Leaderboard Score	Kaggle Private Leaderboard Score
without discounting	0.59803	0.60173
with discounting	0.60818	0.65557

Table 6: The effect of time based discounting over the raw score.

8. Conclusions

The problem to predict *excitement* of a project based on its meta-data and description is hard. The top team on the leaderboard attained AUC scores of around 65. The difficulty of the task could stem from the subjective decision making of the donors at DonorsChoose.org. This can only be captured effectively through a user model learnt through the profiles of these donors.

Not many teams had used all quantitative data from projects.csv. The essay and the project meta-data complement one another. Our ensemble model proves this hypothesis. While other top teams had used weak tree ensembles such as gradient tree boost, our Max-Ent model is simpler and works comparably well. Perhaps the features and its

engineering matter more than the choice of learning algorithm as long as the learning algorithm isn't too simplistic and makes unreasonable assumptions.

References

1. Ratnaparkhi, Adwait. "A maximum entropy model for part-of-speech tagging." *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Vol. 1. 1996.
2. Nigam, Kamal, John Lafferty, and Andrew McCallum. "Using maximum entropy for text classification." IJCAI-99 workshop on machine learning for information filtering. Vol. 1. 1999.