My Solution to the See Click Predict Fix Competition

James Petterson

28/11/2013

Abstract

This document describes my solution to the See Click Predict Fix competition, which aims to quantify and predict how people will react to specific 311 issues.

1 Introduction

It is of the nature of a two month long competition that several algorithms, applied to different feature sets, will be attempted. And if the competition is suitable for ensemble models, it is also expected that the final proposed solution will be a combination of many (if not all) of these attempts.

My solution, therefore, consisted of an ensemble of 24 models built with diverse techniques on various feature sets. Models were built separately for each target variable (number of comments, number of views and number of votes). Only the official data, supplied by Kaggle, was used.

This article is organized as follows: Section 2 explains the variable transformation applied to the labels; Section 3 describes all datasets used with the different algorithms, which are listed Section 4; Section 5 explains how all models were combined, and Section 6 briefly describes the approach taken in the 24-hour hackaton that preceded this competition.

2 Variable Transformation

Since the evaluation measure was RMLSE, all work was done in log space: the labels in the training data were converted with $y \rightarrow log(y + 1)$, and the models' predictions were converted back before submission with $p \rightarrow e^p - 1$. This way the evaluation measure became the more standard RMSE, supported by nearly all off-the-shelf packages.

3 Features

During the course of the competition I attempted building models with different sets of features, preprocessed in various ways. The list below succinctly describes each one of these sets.

- d0:
  - *latitude*, *longitude*, *source*: as in the provided data;
  - *has.descr*: boolean variable indicating whether *description* is not empty;
  - *clean.summary*: *summary* data with similar summaries manually aggregated (via regular expressions) and infrequent summaries combined into a single category;
  - *clean.tag_type*: as above, but for the *tag_type* data;
  - *city*: categorical variable with four levels, computed with hard-coded rules on latitude and longitude;
  - *week*: number of weeks since the earliest data point; the code used to compute this variable had a bug, causing it to be zero for all entries in the test set.
- d1: same as d0, but with *week* fixed and these additional features:
  - *wday*: day of the week;
  - *hour*: hour of the day.
- d1b: same as d1, but with these additional features:
  - *summary.nchar*: number of characters in the *summary*;

  - *description.nchar*: number of characters in the *description*;
  - *summary.nword*: number of words in the *summary*;
  - *description.nword*: number of words in the *description*.

- d1c: same as d1b, but with these additional features:
  - *summary.ncaps*: number of upcase characters in the *summary*;
  - *description.ncaps*: number of upcase characters in the *description*;
  - *summary.nexcl*: number of exclamations (!) in the *summary*;
  - *description.nexcl*: number of exclamations (!) in the *description*.

- d1d: same as d1c, but with missing values in *source* and *clean.tag_type* treated as an additional level (necessary for R's random forests implementation).

- d1e: same as d1d, but with categorical variables limited to 32 levels (also necessary for R's random forests implementation).

- d2: same as d1, but with descriptions converted to vectors of 200 dimensions using word2vec.

- d3: same as d1, but with one hot encoding of categorical features (for linear models).

- d4: same as d3, but with TF-IDF[1] of descriptions (for linear models).

4 Models

The algorithms utilised were the following:

- GBM: Generalized Boosted Regression Models, GBM R package
- VW: Linear Regression with Vowpal Wabbit
- LR: Linear Regression implemented in R
- NN: Neural Networks, nnet R package

[1]Term frequency - inverse document frequency.

- RF: Random Forests, randomForest R package
- Const: constant submission (all 1's)
- KMM: Kernel Mean Matching ([1])

Hyperparameter selection was done by temporally splitting the training data into two periods: 2012-10 to 2013-03 for model training and 2013-04 for model validation. Once the hyperparameters were selected the models were retrained with the concatenation of these two periods (2012-10 to 2013-04) and applied to the test set. Note that all data prior 2012-10 was discarded, except when otherwise noted.

Table 1 list all models that were built. The first column identifies the experiment number; *features* identifies the dataset used; *comments/views/votes* indicates whether a model was built for each one of these target variables (some models were built only for views).

Notes regarding the models (see Table 1):

Note 1: features were split in 5 sets:

- c: *city*
- m: *latitude*, *longitude*, *source*, *clean_tag_type*, *has_descr* and *clean_summary*
- t: *week*, *wday* and *hour*
- d: *description*
- w: word2vec features

The set of features to use was then treated as an hyperparameter (alongside L1 or L2).

Note 1b: same procedure as in note 1a, but with learning rate also treated as an hyperparameter.

Note 2a: a multi-class RF model was built to predict which month each instance of the training data belongs to; this model was then applied to the test data, resulting in a probability distribution of training months[2] for each instance of the test data; these distributions were averaged to obtain an estimative of how much the test data resembles each month of the training data; the instances of the training data were then reweighed according to this distribution, and GBM training proceed as usual.

[2]All training data was utilised, included instances prior to 2012-10.

Table 1: List of models.

			Target Variables			
Id	**Algorithm**	**Features**	**comments**	**views**	**votes**	**Notes**
1	GBM	d0	Y	Y	Y	depth and number of trees as hyperparameters
2	GBM	d1	Y	Y	Y	depth and number of trees as hyperparameters
3	VW	d2	N	Y	N	L2 regularisation, subsets of features (see note 1a)
4	VW	d2	Y	Y	Y	L1 regularisation, subsets of features (see note 1a)
5	LR	d3	Y	Y	Y	L2 regularisation
8	GBM	d1c	Y	Y	Y	depth and number of trees as hyperparameters
12	GBM	d1c	Y	Y	Y	weighted instances v1 (see note 2a)
13	NN	d3	Y	Y	Y	size and decay as hyperparameters
14	RF	d1e	Y	Y	Y	mtry and number of trees as hyperparameters
15	Const	-	Y	Y	Y	
17	GBM	d1c	Y	Y	Y	weighted instances v2 (see note 2b)
19	VW	d2	Y	Y	Y	L1 regularisation + learning rate, subsets of features (see note 1b)
21	GBM	d1c	Y	Y	Y	weighted instances via KMM, sigma 0.01 (see note 3)
22	GBM	d1c	Y	Y	Y	weighted instances via KMM, sigma 1.28 (see note 3)
24	GBM	d1c	Y	Y	Y	weighted instances v3 (see note 2c)
25	GBM	d1c	Y	Y	Y	only latitude and longitude (see note 5)
27	GBM	d1c	Y	Y	Y	only Richmond (see note 4)
28	GBM	d1c	Y	Y	Y	only New Haven (see note 4)
29	GBM	d1c	Y	Y	Y	only Oakland (see note 4)
30	GBM	d1c	Y	Y	Y	only Chicago (see note 4)
35	VW	d2	N	Y	N	only Chicago (see note 4)
36	VW	d2	N	Y	N	only Richmond (see note 4)
37	VW	d2	N	Y	N	only Oakland (see note 4)
38	VW	d2	N	Y	N	only New Haven (see note 4)

Note 2b: same procedure as in note 2a, but this time the training instances (for the multi class RF model) were subsampled as to ensure each month was equally represented in the training set.

Note 2c: a binary classification GBM model was built to predict whether each instance of the data belongs to the training or to the test sets; this model was then applied to the training data to reweigh instances, and GBM training proceed as before.

Note 3: training instances were reweighed using the method described in [1] (KMM).

Note 4: a model was trained with data from only one city; at prediction time instances of the other three cities had predictions fixed as zero.

Note 5: a GBM regression model was built using only *latitude* and *longitude.*

5 Ensemble

All models listed in Table 1 were linearly ensembled using feedback from the public leaderboard. To describe how this was done we will focus in just one model (*comments*), as the procedure is entirely analogous for the other two models.

Assume we have M vectors $p_1, p_2, \ldots, p_M$ with the test set predictions of our target variable (*comments*) for each one of our models. All vectors have size $N \times 1$, where N is the number of test instances (149575). Let's denote by P the $N \times M$ matrix composed of these vectors:

$$P = [p_1 \; p_2 \ldots p_M]$$

where M corresponds to the number of models (24).

Our goal is to find a vector of weights w that, when multiplied by P, minimises the RMSE loss of the resulting ensemble. If we call y the vector with the ground-truth, this translates to:

$$w = \underset{w}{\operatorname{argmin}} (y - Pw)^T (y - Pw) + \frac{1}{2}\lambda \left\| w \right\|^2$$

where we added a regularisation term to avoid overfitting. This is a standard linear regression problem, with closed form solution:

$$w = (P^T P + \lambda I)^{-1} P^T y \tag{1}$$

$P^T P$ is known, and we can estimate $P^T y$ as follows:

- Make a submission of all zeros[3]; this will return

  $$s_0 \approx \sqrt{\frac{1}{3N}\left(y^T y + y'^T y' + y''^T y''\right)}$$

  where y' and y'' are the ground truth labels for the other two models (*views* and *votes*); this is an approximation, as the leaderboard score is based on 30% of the data, and assumes that the split of instances for the public and private leadearboards was random;[4]

- Make a submission of p_1 (the first model), but keeping only the predictions for *comments*, and replacing the predictions for *views* and *votes* with zeros; this will return

  $$s_1 \approx \sqrt{\frac{1}{3N}\left[(y - p_1)^T (y - p_1) + y'^T y' + y''^T y''\right]}$$

- Take the difference of the squares of s_0 and s_1 to estimate $p^T y$:

  $$\begin{aligned} 3N(s_0^2 - s_1^2) &\approx 2p_1^T y - p_1^T p_1 \\ p_1^T y &\approx \frac{3N(s_0^2 - s_1^2) + p_1^T p_1}{2} \end{aligned}$$

- Repeat the previous two steps for the other models $(p_2, p_3, \ldots, p_M)$ to obtain $p_2^T y, p_3^T y, \ldots, p_M^T y$, and combine the results to solve equation (1).

This procedure is similar to what was described in Section 7 of [2], except that in that case there was only one target variable.

My final model was an ensemble of all 24 models with λ set to 1, achieving a private leaderboard score of 0.28994.

[3]This is actually one of the benchmarks provided by Kaggle.

[4]See [2] for an estimation of the degree of overfitting due to this estimation.

6 Hackaton

My solution to the 24-hour See Click Predict Fix Hackathon was a subset of what is described here:

- only two data sets were used (d1 and d2);
- only two models were built: GBM and VW.

The ensemble procedure was exactly the same.

References

[1] J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Scholkopf. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, volume 19, page 601. Citeseer, 2007.

[2] A. Toscher and M. Jahrer. The BigChaos Solution to the Netflix Grand Prize. 2009.