Predicting the urgency of urban issues

Christian Braz
George Washington University
Department of Data Science

Abstract

City maintenance is laborious and expensive. One of the main challenges is to monitor unpredictable situations like potholes, grafiti, broken footpaths, and so on. Once detected, government officials must be able to allocate the resources timely and prioritizing the issues most relevant for the citizens. The pervasive presence today of mobile internet connection in urban centers has enabled modern ways of interaction between the municipality and the population, resulting in the so-called Government 2.0. One of such way is crowdsourcing platforms, such as See-Click-Fix, FixMyStreet, CitySourced, OpenIDEO, and many others, which allow and stimulate collaborative participation by reporting urban issues. The importance of an issue can be endorsed via votes in the platform, meaning that issues with more votes represent the overall felling of the neighborhood that those should be solved first. This, in turn, constitutes important information to help organizing logistics, allocate resources, and fulfill citizens' well-being feeling. The problem is that may take time until collecting enough votes to be able to estimate the urgency of an issue. In this work we propose to estimate the number of votes an issue will receive using machine learning techniques. As the number of votes is a proxy to the urgency, we hope to improve city maintenance by providing in advance sensitive information.

Keywords: crowdsourcing, government 2.0, web 2.0, machine learning

Introduction

Since its rising in early 90s, the World Wide Web has been modifying the way people interact. Its distributed infrastructure, built upon Internet's top layers, made it pervasive and an ideal tool to enable all kinds of communications services. On a business perspective, new jargons were created trying to categorize such virtual interactions, such as Business-to-Consumer, Business-to-Bussiness, and Customer-to-Customer (Brzozowska, 2018). Social networks are also an example of interaction. Facebook, Twitter or Instagram are well known interactive platforms which enable users express their opinions. Governments have also been experimenting changes in the way the citizens interact with them and Kanhere (2011) provides a comprehensive overview. The so-called Government 2.0 implies that information should flow not only from the government to the citizens but also from citizens to the government and among citizens.

City maintenance is expensive, since it involves monitoring and fixing a variety of complex issues related to public safety, environmental problems, and quality of life. In particular, monitoring unpredictable urban issues (e.g., potholes, damaged street signs, graffiti, street light issues, damaged trees) usually requires a large number of employees working on a permanent basis. However, these same urban centers are full of people armed with their GPS enabled cell phones, and represent an important asset to help fine-grained monitoring capabilities. Civic engagement platforms, such as See-Click-Fix.com, allow citizens to report urban issues by entering, for instance, GPS location and free text describing the problem. Having this information, in turn, helps municipalities to reduce costs and improve logistics by better allocating resources.

Another important aspect in city maintenance, is how to rank the problems. As the resources are limited, most of the time would not be possible to fix all the open issues. Thus, it is necessary to have a way to measure which ones are more important. In the same crowdsourcing platforms, the importance of an issue can be endorsed via votes, meaning that the number of votes can act as a proxy to the urgency of an issue. Therefore, a rank of issues can be built based on those votes. While the number of votes an issue receives ultimately reflects its urgency to be solved, acquiring a significant number of votes may take several days or weeks, leading to ineffectiveness and late responses. In order to become more responsive and better meet the needs and concerns of the citizens, government officials must be able to prioritize more urgent issues as soon as possible.

In this work, we evaluate machine learning techniques to predict the number of votes an issue will receive. More specifically, we evaluate Ridge Regression, Lasso Regression, Support Vector Machine (SVM), and Long Short Term Memory (LSTM) algorithms. We hope to demonstrate the feasibility of this approach, and help to contribute in modernizing an important aspect of public administration. This wok is organized as follows: next we expose more details about the problem and the data, then we describe the experiment and show the results, and finally we address our final considerations.

Problem statement and the data

Our experiments use the [SeeClickFix] hackathon data, comprising 34 Mb of size and consisting of a total of 223,129 reported issues from four cities: Oakland, Richmond, New Haven, and Chicago. SeeClickFix.com is a crowdsourcing initiative that allows citizens to report issues categorized in 311 different types. The main way to submit a request is taking a picture and sending it via their mobile app. They claim that the service manages the whole work-flow, linking the citizen with the city hall.

Table 1 shows a description of the data. There are eleven attributes which some of them having a high rate of missing values. Figure 1, 2, and 3 show the distribution of the records among issue categories. Issues related to trash, trees and potholes correspond to more than 50% of all issues, whereas issues related to lost and found items, or to public art, are rarely reported by citizens. Figures 1 and 2 show the absolute number of reports and votes grouped by issue type. Not always the most frequent issue is the most voted one. Figure 3 gives an insight about the urgency of an issue by computing the ratio votes:occurrences. We can see that issues like drug dealing and public concern, although

Column	Description	Type	% Missing Values
id	randomly assigned id	Numeric	0
latitude	latitude of the issue	Numeric	0
longitude	longitude of the issue	Numeric	0
summary	short text title	Text	0
description	longer text explanation	Text	52
num_votes	number of user votes	Numeric	0
num_comments	number of user comments	Numeric	0
num_views	number of views	Numeric	0
source	where the issue was created	Categorical	13
created_time	time the issue originated	Timestamp	0
tag_type	type of issue	Categorical	76

Table 1
Dataset summary

Feature	Max	Min	Avg
Summary	49	1	6.4
Description	924	1	13

Table 2
Statistics of the summary and description fields

rare, receive lots of votes when occur, revealing the natural concern citizens have about them.

Another important statistic is the length of the summary and description fields. This has influence in how to set the parameters of our models and are depicted in Table 2. The average number of words for summary are 6.4 words, and for description are 13 words.

The problem as originally described in the hackathon, is to predict num_votes, num_views, and num_comments. The evaluation metric defined for the competition, Root Mean Squared Logarithmic Error (RMSLE), is described by the following equation:

$$RMSLE = \sqrt{\frac{1}{n}\sum(log(p_i + 1) - log(a_i + 1))^2}$$

where p and a are arrays of shape (num_samples, num_targets).

As we mentioned before, our main goal is to help in estimating the urgency of a reported issue by predicting num_votes. Therefore, in this work we concentrate in just predicting it. However, we stick with the defined loss function to enable, eventually, future comparison with results reported in the competition.

In the next section we discuss in more detail our experiments. We make a brief description of the algorithms and features we use, and also show the RMSLE issued by each of them.

Experiment and results

In order to evaluate the feasibility of our proposal, we test our hypothesis using four different supervised machine learning algorithms designed to solve regression problems. The

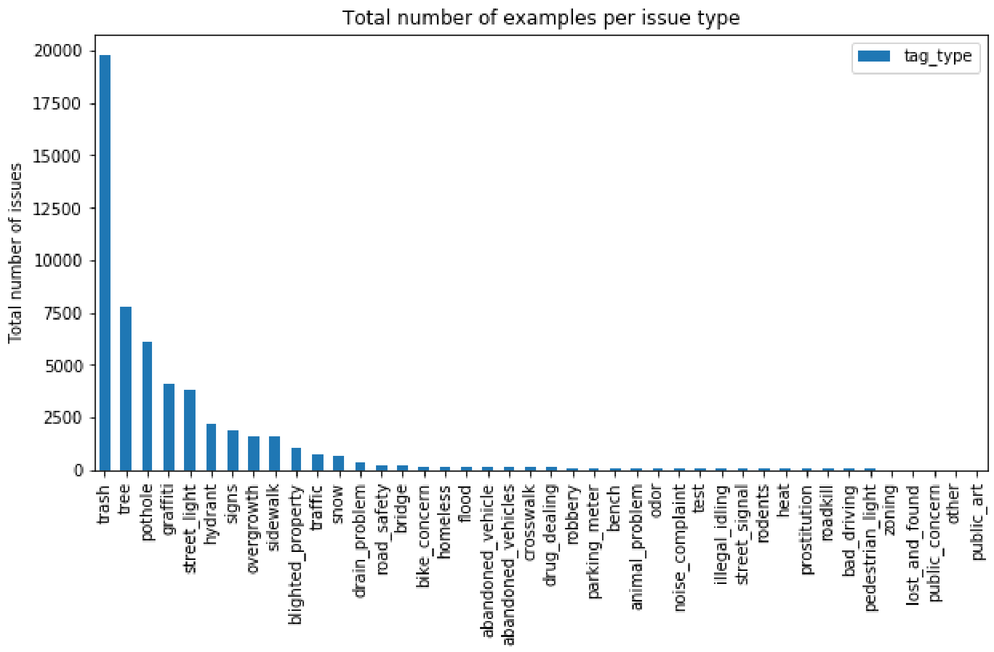

Figure 1. Total number of records per issue category

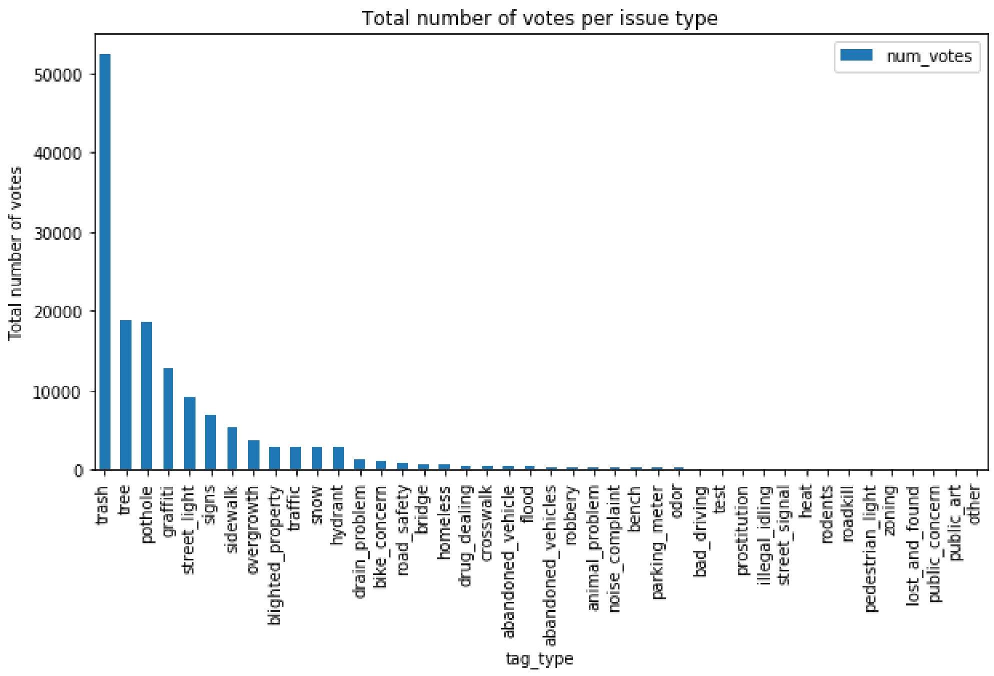

Figure 2. Total number of votes per issue category

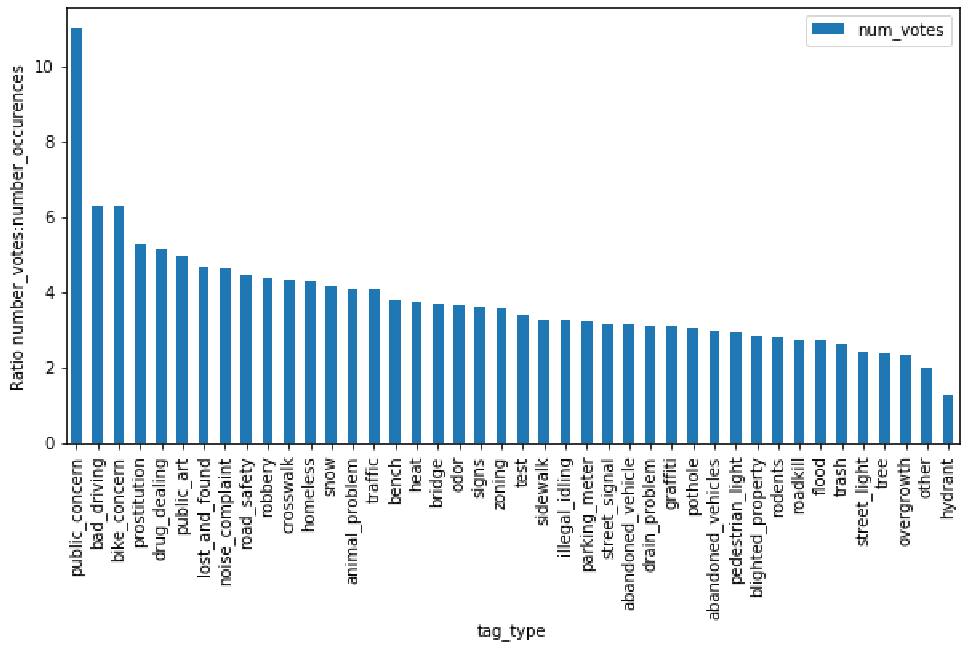

Figure 3. Ratio between number of votes and number of issues

overall methodology for the experiment can be summarized by the following points:

• Extract suitable features: the features we use to train our models are description, summary, source and location (latitude and longitude). We divide the experiment in two parts:
• Part A: using only the textual features: summary and description.
• Part B: using all four features: summary, description, source and location.

We do that because we do not know whether or not the features *source* and *location* would be available in other crowdsourcing platforms. We make the assumption that textual description features will always be present. Having both results separately helps to find which model is better in a more general scenario.

• Configure the parameters and train the models using a training set.
• Test the performance of the model in a test set.

This section is organized as follows: first we describe in more details how we use the selected features, then we make a brief description of each algorithm used, and finally we present the results obtained for Part A and B.

Feature Extraction

One important matter in using supervised learning algorithms is to have data in a suitable form to feed them. The task of transform the raw data in features is called feature extraction and some of the representation techniques we use are:

• Bog-of-words (BOW): This is the most common approach to extract features from text. The idea is to represent each document as a (sparse) vector where each cell counts the number of times a particular word has occurred in the text. BOW does not consider the order of words and it is often used with n-gram models.

• Word Semantic Vectors: word vectors, Word2Vec, word embedding are all designations of how has been known the recent supervised learning technique to learn distributed representation of words Mikolov, Sutskever, Chen, Corrado, and Dean (2013). The main idea of the skip-gram with negative sampling algorithm is to train a two layer neural network to predict the probability of all other words in the vocabulary be in the vicinity of this word. At the end of the training, the hidden layer will be the word vector because one way for the network to output similar context predictions is if the word vectors are similar. So, if two words have similar contexts, then the network is motivated to learn similar word vectors for these two words.
The transformations we apply in the features are:

• prediction and summary: to use with Lasso, Ridge and SVM, transform to a BOW representation with 5000 words vocabulary size and pruning words that occur in more than 50% of the documents. To use with LSTM we convert each word in a 200 dimensional word embedding. Fill with a blank character whenever it is null.

• source: remove the examples where it is empty, leaving a total of 194,278 records. As it is categorical, convert to its one-hot-encoding representation.

• location: as they are latitude and longitude values represented as decimal degrees, scale and round to three decimal places which corresponds to neighborhood precision.

Algorithms

Follow a brief description of the algorithms we use:

• Ridge regression: is a variant of the Ordinary Least Square that applies a penalty to the loss function in order to prevent from overfitting. This is achieved by adding an extra term to the loss that is a squared sum of the weights. This stimulates the optimization procedure to shrink the weights parameters (but not zeroing them). It is also known as L2 regularization.

• Lasso regression: is also known as L1 regularization. The penalty it applies is a sum of the absolute values of the weights. This leads to a different effect compared to the Ridge method as the weights can be set to zero if they are not relevant. Therefore, Lasso also acts as a feature selection mechanism.

• Support Vector Machines (SVM): is a flexible and powerful algorithm based on the idea of “large margin”. It builds hyperplanes which can be used for classification or regression. These hyperplanes are built in such a way to preserve the larger distance possible between the samples contained in the two sides being separated. This in turn implies in lower generalization error. SVM is essentially a linear model. However, it can be extended to non-linear boundaries by using different kernels which implicitly transform the original features using non linear functions.

• Long Short Term Memory: is a class of neural models designed to tackle sequence like problems, i.e., problems where past information is relevant to compute actual outcomes. Among common problems defined as sequential are speech recognition, time series analysis, and neural language models. LSTM is an improved version of a simpler neural model called Recurrent Neural Network (RNN), Figure 4 left of Jurafsky and Martin (2000). They share the idea of having a time component, a memory, to enable them to take into account history of information. LSTM is more robust and complex and is the *de facto* solution as it allows to process and learn from longer sequences by solving many RNN limitations. The main

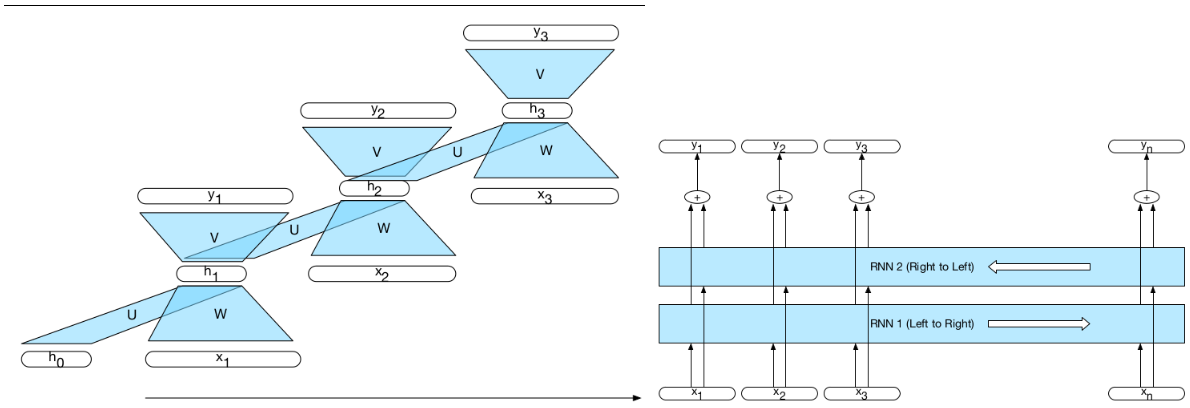

Figure 4. Left: An schematic view of a simple recurrent neural network shown unrolled in time. Right: A bidirectional RNN.

drawback of simple RNN is that it uses a single matrix of weights (the memory component) to compute actual predictions and also to accumulate past information. Another problem related to neural models in general is the vanishing or exploding gradients, an intrinsic effect associated with the way these models learn and that is leveraged in sequence-like problems as the sequences trend to be lengthy. LSTM tackle these issues using an intricate mechanism of gates to control the extent to which a new value flows into a LSTM unit (input gate), the extent to which a value remains in the unit (forget gate), and the extent to which a value in the unit is used to compute its output (output gate).

- Bidirectional LSTM: Figure 4 right shows a scheme of a bidirectional RNN. In a regular RNN (or LSTM) at each step in time the network learned everything about the sequence up to that point. We say that the model is taking into account the context in its left. For many problems, however, we have access to the entire sequence beforehand. It turns out that it is also valuable and possible to keep track of the context in the right as well. Then, the network can combine both contexts and, as a result, modeling a better representation of the data. This is accomplished by a bidirectional RNN which process the input sequence from the start to the end, and also in reverse, terribly capturing left and right contexts into a single representation.

Experiments and results

In this section we describe the experiment and the result for each one of the algorithms experimented. The experiment is divided in two parts as we want to assess the feasibility of this approach in a more general setting. Therefore, Part A addresses only textual features describing the problem. We are assuming that a textual description of the issue is common among different crowndsourcing platforms. In Part B we also use the location and source as features, as they are available in SeeClickFix.com data.

Ridge Regression. We use the implementation available in `sklearn.linear_model.RidgeCV` which performs a built-in cross-validation to find the best value for the regularization parameter. The RMSLE for Part A is 0.196 and for Part B is 0.176.

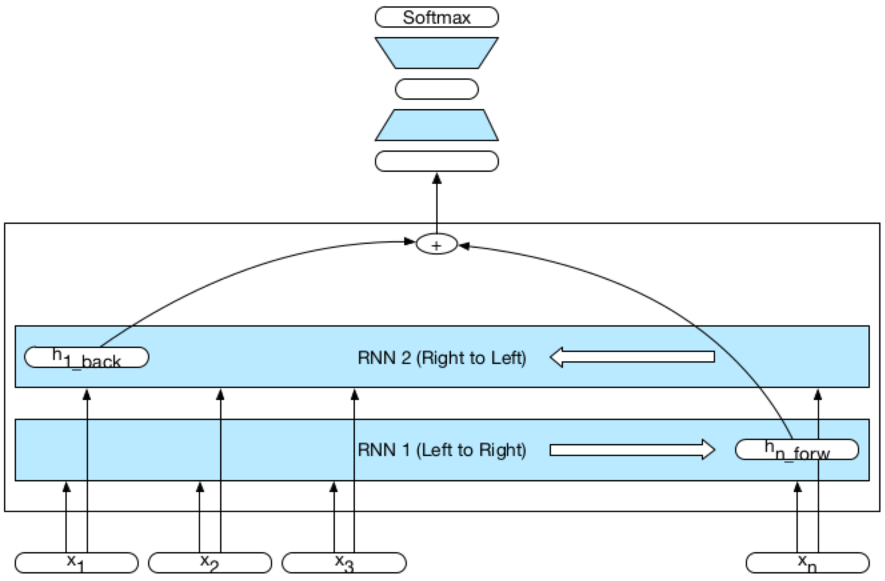

Figure 5. A bidirectional RNN for sequence classification.

Lasso Regression. We use the implementation available in `sklearn.linear_model.LassoCV` which performs a built-in cross-validation to find the best value for the regularization parameter. The RMSLE for Part A is 0.201 and for Part B is 0.180.

SVM. We use the implementation available in `sklearn.svm.LinearSVR` which is an efficient implementation of the SVM with a linear kernel. Although SVM allows other non linear kernels, according to the scikit-learn documentation this is practical only when the number of samples and features are reduced, which is not our case. We employ grid-search to find the best value for the regularization parameter C=0.001. The RMSLE for Part A is 0.174 and for Part B is 0.166.

Bi-LSTM. We build a multi-input sequence model in Keras (https://keras.io) to predict the number of votes. An sketch of a Bi-LSTM used to sequence classification can be seen in Figure 5 from Jurafsky and Martin (2000).

The RMSLE for Part A is 0.168 and for Part B is 0.147.

Table 3
Accuracy and F1-score

Algorithm	Feature	Acc	F1
Logistic Regression	BOW	0.726	0.775
Logistic Regression	Custom Word2Vec	0.762	0.820
SVM (C=1,linear)	Custom Word2Vec	0.767	0.823
Logistic Regression	Glove Word2Vec	0.783	0.834
SVM (C=1,linear)	Glove Word2Vec	0.783	0.834

Conclusion

In this work we evaluated two machine learning algorithms for classifying sentiment in tweets messages. We used word vectors, which instantiate the linguistic idea of distributional semantic, as features for the classifiers, and compare their performance with the traditional approach of bag-of-words. We used two different sets of word vectors, one that we trained and called them custom word vectors, and word vectors from the Glove project. To train the classifiers, we carefully concatenated five different manually labeled twitter corpus used in previous sentiment classification experiments.

For future experiments, we would recommend tests with higher dimensional vectors, different classifiers, and ensemble techniques for improving accuracy in a similar training approach as used here. Also, tests with neural sequence model as Recurrent Neural Networks are highly desirable as these models have been issuing the state of the art accuracy in many sequence dependent tasks.

References

Brzozowska, A. (2018). E-business as a new trend in the economy. *Procedia Computer Science*, *65*, 1095 - 1104.

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition* (1st ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th international conference on neural information processing systems - volume 2* (pp. 3111–3119). USA: Curran Associates Inc. Retrieved from http://dl.acm.org/citation.cfm?id=2999792.2999959