Twitter Analysis of Road Traffic Congestion Severity Estimation

Sakkachin Wongcharoen and Twittie Senivongse
Department of Computer Engineering, Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
sakkachin.w@student.chula.ac.th, twittie.s@chula.ac.th

***Abstract*—Road traffic congestion is one of the problems in large cities. While Twitter has become a major means for communication and message dissemination among Internet users, one question that arises is whether Twitter messages alone can suggest road traffic congestion condition. This paper proposes a method to predict traffic congestion severity level based on the analysis of Twitter messages. Different types of tweets data are used to construct a C4.5 decision tree model for prediction, including tweets from selected road-traffic Twitter accounts, tweets that contain road-traffic-related keywords, and geo-tagged tweets whose volume suggests large crowds in certain areas. In the training that used tweets data of one of the top congested roads in Bangkok, density of tweets has high information gain in classifying congestion severity level. Via a web application, the model can provide a traveler with an estimate of the congestion severity level of the road at every 30 minutes. Such advance congestion information can help the traveler to make a better travel plan. In an evaluation using 10-fold cross validation, the model also shows good performance.**

Keywords-Twitter; traffic congestion severity; decision tree

I. INTRODUCTION

Nowadays social networking services are widely used by Internet users for communication, e.g. sharing information and personal status, publicizing events, news reporting etc. Twitter, Facebook, and Instagram are some of the popular social networking services, each providing APIs for retrieving social network data. By nature, Twitter messages [1] are more public and come in large volume with high velocity. Therefore, via the easy-to-access APIs, Twitter messages become a large data source for analysis of socially valuable information, such as social behavior, current public interests, public opinions on particular topics, occurrence of events etc.

As road traffic congestion is one of the main problems in large cities including Bangkok, road traffic information is made available through several authorized sources (e.g. government agencies, traffic reports on radio and television channels, traffic websites, and map services). Those mainly use traffic data that are captured at the locations from traffic cameras and report the current condition of the traffic. Since Twitter users tweet to spread information, we aim to answer one question, i.e. can Twitter messages alone suggest road traffic congestion condition? Our assumption is that different types of tweets data might be useful to predict congestion severity: 1) There are a number of Twitter accounts that report current road traffic condition, i.e. @js100radio, @fm91trafficpro, @traffy, @weather_th, and @longdotraffic, 2) Some users tweet to tell their followers about current road traffic congestion condition, either to share or even to complain about the condition, and 3) Some Twitter users check-in their current locations when they go to different places, and the volume of these geo-tagged tweets can indicate large crowds, and effectively current traffic volume, in certain areas. Such Twitter usage motivates us to study whether tweets alone can be used to construct a prediction model for congestion severity. Specifically, if the model can predict congestion severity level of a road at some point in the future, rather than at current time, a traveler could better plan his/her travel. If the traveler knows that the traffic will be, for example, less congested in half an hour, he/she could wait until then before setting off on a drive.

This paper proposes a method to predict traffic congestion severity in the next 30 minutes using tweets from selected accounts, tweets containing traffic-related keywords, and geo-tagged tweets. The prediction model is a C4.5 decision tree that was trained by tweets data of one of the top congested road in Bangkok (i.e. Pahonyothin Road). The attributes of the training data are: 1) day of week, 2) hours of day, 3) minutes of hour, and 4) tweets density. Via a web application, a user can view an estimate of the congestion severity level of the road at every 30 minutes. We user 10-fold cross validation to evaluate the performance of the model.

The rest of the paper is organized as follows. Section II reviews a number of related work. Section III presents the prediction method, followed by the experimental result and evaluation in Section IV. Section V discusses some limitation of the method, and the paper concludes in Section VI.

II. RELATED WORKS

Twitter messages have been analyzed for various purposes. Related examples are as follows.

A. *Rapid Estimate of Ground Shaking Intensity by Combining Simple Earthquake Characteristics with Tweets [2]*

Burks et al. used a combination of earthquake characteristics and geo-tagged tweets that contained keywords

978-1-5090-2033-1/16/$31.00 ©2016 IEEE

978-1-5090-2033-1/16/$31.00 ©2016 IEEE

about earthquakes and tsunami and occurred within the first 10 minutes following Japanese earthquakes to rapidly estimate ground shaking intensity. They constructed different regression models and found that the best one had the number of tweets within a certain radius of the earthquake as one feature. We adopt the idea and use tweets density within a certain radius of a road as one feature in our model.

B. *Predicting Severity of Road Traffic Congestion Using Semantic Web Technologies [3]*

Lécué et al. proposed a method to predict severity of road traffic congestion in Dublin city. They used data from various sources, i.e. road weather condition, weather information, Dublin bus stream, social media feeds, road works and maintenance, and city events. These data were represented as OWL knowledge base and rules were encoded in SWRL for reasoning about traffic congestion. Regarding social media feeds, they used only tweets from reputable Twitter accounts of road traffic condition. Contrarily, we study the usefulness of Twitter on its own and use different types of tweets.

C. *Social-Based Traffic Information Extraction and Classification [4]*

Wanichayapong et al. applied syntactic analysis to determine structure of road-traffic-related texts in tweets and classified them before retweeting. The work captured tweets from all accounts, tokenized them, and kept only those that contained traffic or accident keywords and did not contain banned words (such as rude or question words). Tweets were classified as either point messages (reporting traffic condition at a specific location) or link messages (reporting traffic condition between two points). After that, geolocation was added to the tweets. We adapt from the steps taken by this work to extract road information from tweets but we also consider geo-tagged tweets that may not contain traffic keywords but are tweeted within a certain radius of the road.

D. *Traffic Prediction Models for Bangkok Traffic Data [5]*

Klakhaeng et al. presented two prediction models for Bangkok traffic data using C4.5 decision tree. The traffic condition data that were used to train the models were the data of one link (link 1206) of a congested road in Bangkok, and obtained from processing the video streams from CCTV cameras. The first model could predict traffic congestion condition in the next 30 minutes using the attributes 1) day of week 2) hours of day 3) minutes of hour 4) time units of five minutes starting from 12:00 am 5) current congestion condition, 6) current congestion condition of the inbound road(s), and 7) current congestion condition of the outbound road(s). The second model could predict the change of congestion condition in the next 20 minutes (as based on their study, the traffic condition would change to more congested or less congested in 20 minutes). We are motivated the most by this work. We also use C4.5 decision tree but are interested in Twitter-based prediction

III. CONGESTION SEVERITY PREDICTION METHOD

The overview of the traffic congestion severity prediction method is depicted in Fig. 1. The method consists of 4 steps, i.e. 1) data collection which collects road data and Twitter messages, 2) model construction which prepares training data and trains a decision tree model, 3) model evaluation which evaluates prediction performance of the model, and 4) model presentation which uses the model in the implementation of a web application to report congestion condition. The following subsections and Section IV describe the details of the steps.

A. *Data Collection*

We collected data from three sources, i.e. road data, Twitter data, and actual road traffic congestion data. They were processed and stored in a MongoDB database as in Fig. 2.

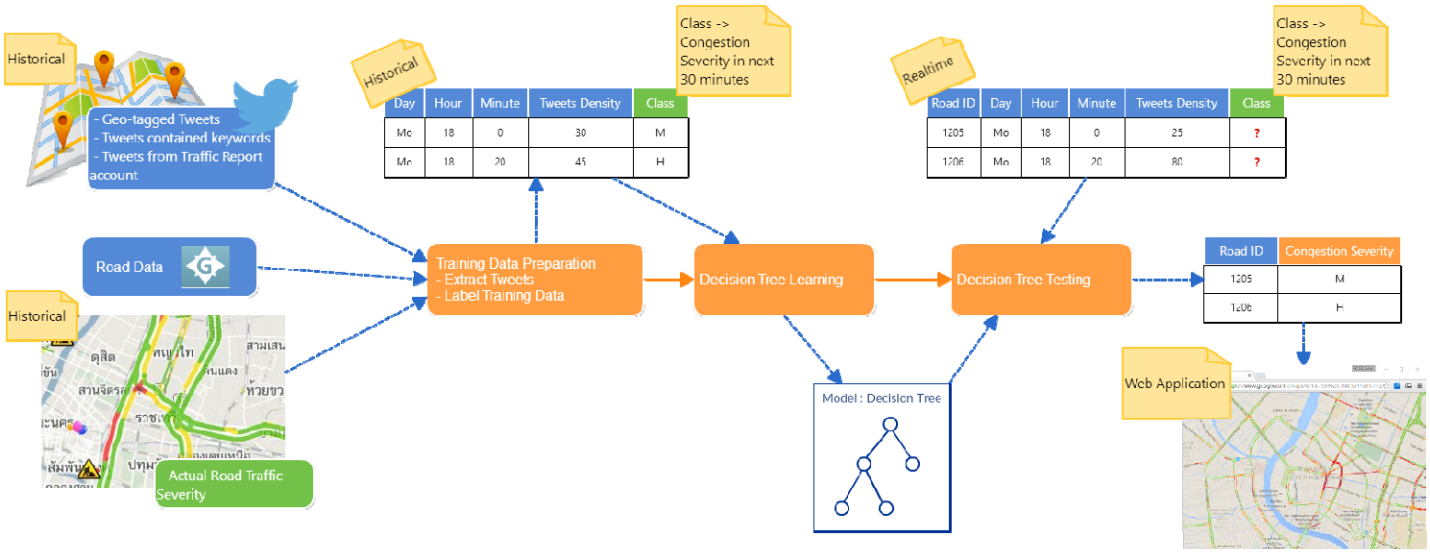

Figure 1. Overview of the method.

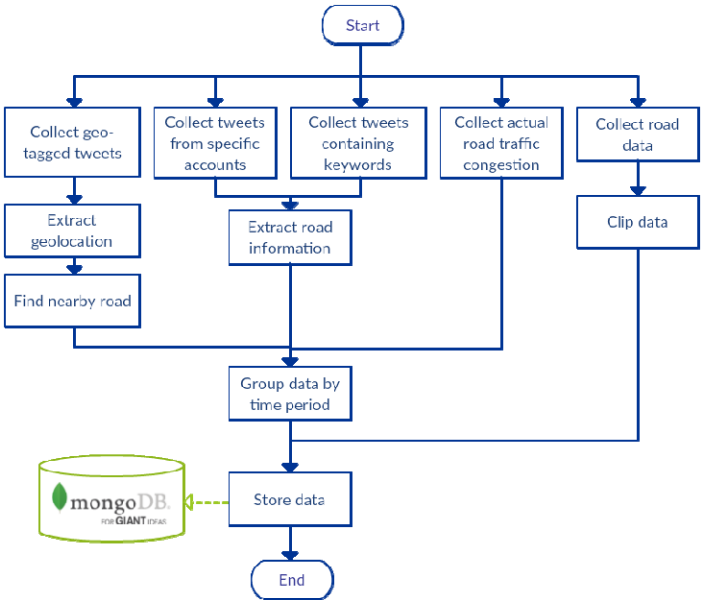

Figure 2. Data collection.

1) Road Data

First, we collected road data from Geofabrik [6] which provides Thailand OpenStreetMap data. Then, we used QGIS [7] application to select (or clip) only *primary, secondary,* and *motorway* types of roads in Bangkok, using the spatial query tool of QGIS, as these three road types are the types of the actual road traffic congestion data. We exported the selected data to the GeoJSON format [8] and then stored them in a MongoDB collection [9]. Finally, we created *2dsphere* index in the geometry field of the collection. This collection is called the *Road collection*.

2) Twitter Data

We wrote three python scripts to collect tweets via Twitter Streaming APIs [10]. We collected data for 30 days during 5 January – 5 February 2016, and then processed and stored them in MongoDB.

- To collect geo-tagged tweets in Bangkok boundary, first, we used the *statuses/filter* endpoint of the API. We passed the values *'100.329071, 13.49214, 100.938637, 13.95495'* which represent Bangkok boundary as the *location* parameter to the endpoint. We collected 1,095,054 tweets in this step. Second, we filtered out tweets that were automatically tagged by the devices because the tag is polygon and we could not find suitable reference centroid from that. Third, we extracted geolocation from each tweet by getting the *coordinates* property of the tweet. Next, we used the *$near* operation of MongoDB to find which road from the *Road collection* is within 1,000 meters of the tweet geolocation. We used only the tweets that we could find nearby roads, so there were 328,204 tweets remaining. Then, we extracted the *timestamp_ms* property of each tweet to get its hour and minute. We put tweets whose minute value is in [00, 29] in one group and [30, 59] in another group, because our actual road traffic congestion data were recorded at every 30 minutes. For example, tweets whose timestamps were 10:00, 10:05, 10:16, and 10:29 would be considered as tweets with the timestamp 10:00 whereas tweets whose timestamps were 10:30, 10:35, 10:46, and 10:59 would be considered as tweets with the timestamp 10:30. Finally, we stored these processed tweets in a MongoDB collection called the *Geo-tagged collection*.
- To collect tweets from the selected Twitter accounts, first, we used the *statuses/filter* endpoint of the API. We passed the Twitter account IDs of @js100radio, @fm91trafficpro, @longdotraffic, @weather_th and @traffy, which report traffic information to their followers, as the *follow* parameter to the endpoint. We collected 65,520 tweets in this step. Second, we used Lexto [11] to tokenize texts in tweets in order to extract road information. Since Lexto uses a dictionary to perform its task, we appended road names from our *Road collection* to the default dictionary of Lexto. We used only the tweets that we could find related road names, so there were 3,417 tweets remaining. Next, we grouped these tweets by their timestamps for every 30 minutes in the same way as we did with the geo-tagged tweets. Then, we stored these processed tweets in a MongoDB collection called the *Follow collection*.
- To collect tweets containing keywords related to traffic congestion, first, we used the *statuses/filter* endpoint of the API. We passed the values "*รถติด, อุบัติเหตุ, ถ., ถนน, จราจร, รถเยอะ, ติดขัด*" (i.e. *traffic jam, accident, rd., road, traffic, jam, congest*) as the *track* parameter to the endpoint. We collected 12,293 tweets in this step. Next, we extracted road information and grouped these tweets by their timestamps for every 30 minutes in the same way as we did with the geo-tagged tweets. We used only the tweets that we could find related road names, so there were 1,821 tweets remaining. Then, we stored these processed tweets in a MongoDB collection called the *Keyword collection*.

3) Actual Road Traffic Congestion Data

We got the actual traffic congestion severity data from The Intelligent Traffic Information Center Foundation (iTIC) and Longdo Traffic. (These data would be used later to label the classes of the training data.) Congestion severity is categorized into three levels, i.e. L (low), M (medium), and H (high). Note that the actual road data are finer-grained than the road data in our *Road collection* because the actual congestion data were collected for "links" (or sections) of roads, whereas the road information in our *Road collection* were collected for "roads" since traffic information in tweets mostly were associated with roads, rather than links of roads. Also, the actual congestion data for the links were collected at every 30 minutes. We put these actual data in a MongoDB collection called the *Congestion collection*.

B. Model Construction

To construct the model, we prepared the training data from all the collected data in Section III.A. An example of the training data to predict congestion severity level (H, M, L) are

shown in Table I. There are four attributes, i.e. 1) day of week 2) hours of the day 3) minutes of the hour, and 4) tweets density.

As described previously in Section III.A, we extracted the *timestamp_ms* property of each tweet and put the tweets in groups by their timestamps. For example, the first group (or first record) of the training data in Table I refers to the tweets that occurred on Tuesday during 8:00-8:29 (or H8:M0) whereas the third group refers to the tweets that occur on Tuesday during 2:30-2:59 (or H2:M30). Note that "H" was appended to hours of day and "M" to minutes of hour to make them nominal values. The tweets in all three collections (i.e. *Geo-tagged*, *Follow*, and *Keyword collections*) which belonged to each group then were counted, as shown in Fig. 3, to obtain tweets density of the group. To do so, we assigned different weights for the tweets from different collections based on how strong they represent traffic information. That is, tweets from the selected Twitter accounts concern actual traffic condition, whereas the geo-tagged tweets are merely tweets with geo-tags and may not be about traffic condition. Additionally, tweets with traffic-related keywords may or may not give information about congestion condition. We hence assigned the maximum weight of 10 to each tweet in the *Follow collection,* medium weight of 4 to each tweet in the *Keyword collection,* and minimum weight of 1 to each tweet in the *Geo-tagged collection*. Then, we counted tweets density of each group by calculating the total weights of the tweets in the group. For example, if there was one tweet from each of the three collections which belong to the same group Mon H8:M0, tweets density for this group would be 15. We selected a road with the highest tweets density for training, i.e. Pahonyothin Road, in this case. All tweets in each group were merged and stored in the *Merged collection*.

Next, to label a class of congestion severity (H, M, L) to a group of tweets, the process is shown in Fig. 4. For each group of tweets, we queried congestion severity of Pahonyothin Road from the *Congestion collection* using day of week, hours of day, and minutes of hour of the next 30 minutes after the timestamp of the group, as a query condition. For example, for the group Tue H8:M0 (8:00-8:29), we queried the congestion severity of Tue during 8:30-8:59. Since Congestion collection stored congestion severity level of each link of the road, so the query returned several congestion severity levels for different links of the road during 8:30-8:59. Therefore we had to find a representative value to represent the congestion severity level of the whole road. As the severity level is in a nominal scale, we chose the median value. After that, each group with a labeled class was stored in the .csv file of the training data.

TABLE I. EXAMPLE OF RECORDS OF TRAINING DATA (.CSV FILE)

Day	Hour	Minute	Tweets Density	Class
Tue	H8	M0	71	H
Tue	H6	M0	23	M
Tue	H2	M30	3	L

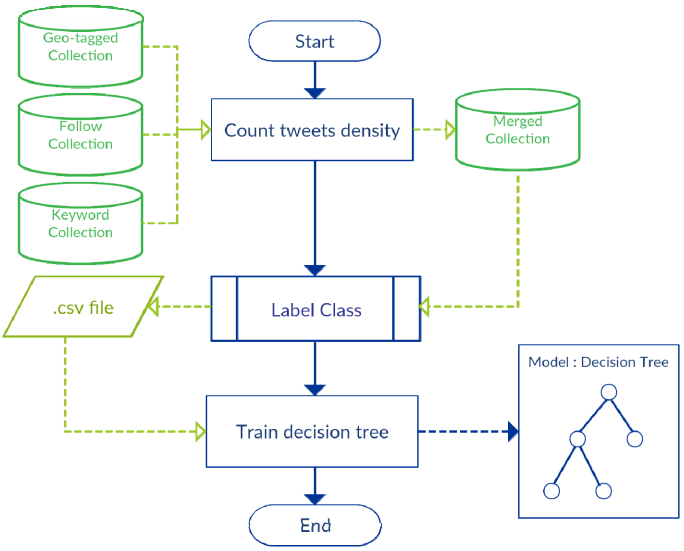

Figure 3. Training data preparation and model construction.

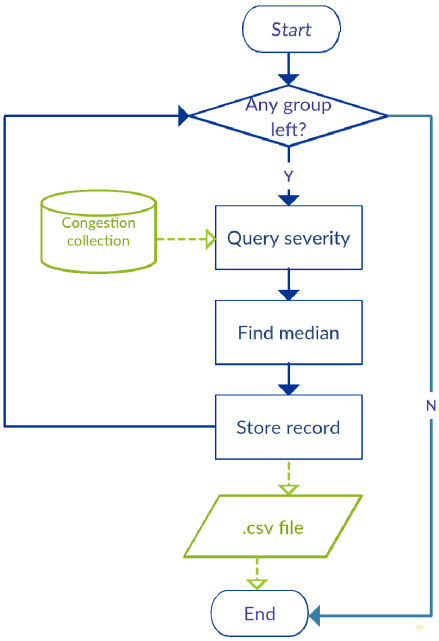

Figure 4. Labeling training data with congestion severity class.

After we got the .csv file, we used the Weka 3 tool [12] to construct a C4.5 decision tree model. Since Weka 3 implements C4.5 in its J48 decision tree, so we used J48 pruned tree for our model. When Weka 3 finished training the data, we evaluated the model using 10-fold cross validation (see Section IV). After that, we exported the result as a model file to use with our web application.

C. Use of Model

To use the model, we developed a web application that can estimate congestion severity level of the road. The Django web framework [13] was used with the Python-Weka wrapper

library [14] to deal with the model file from the previous step. Fig. 5 describes the process of the web application which comprises two parts, i.e. backend and frontend.

- For the backend design, we create scheduled tasks to collect tweets from selected accounts, tweets containing traffic-related keywords, and geo-tagged tweets, and process them in the same way as the training data. While the scheduled tasks work, we inspect tweets density for each road. If any road has its tweets density equal to 10, we will test its attributes with our model and then store the predicted result in a MongoDB collection for the frontend to display. (Note that this tweets density threshold is defined to reduce tweets processing.) We use the Python-Weka-wrapper library to call necessary Weka methods for prediction. The prediction will be executed every 35 minutes because the model can predict the congestion severity in the next 30 minutes, plus 5 minutes for checking tweets density with the threshold.
- For the front end design, we offer an option for a user to specify a road name. If a road name is specified, the web application will display only the predicted congestion severity for that road. But if the user does not specify a road name, the application will display congestion severity of all roads whose tweets density values meet the defined threshold described earlier in the backend design part. Also, the web application will be refreshed every 35 minutes if the user does not terminate the application.

Fig. 6 shows the UI of the web application. Roads data and their congestion severity estimation results are displayed on map controls whose base map is Google Maps, in the *Map* section. Different colors represent different severity estimation results, i.e. red for "H", yellow for "M" and green for "L".

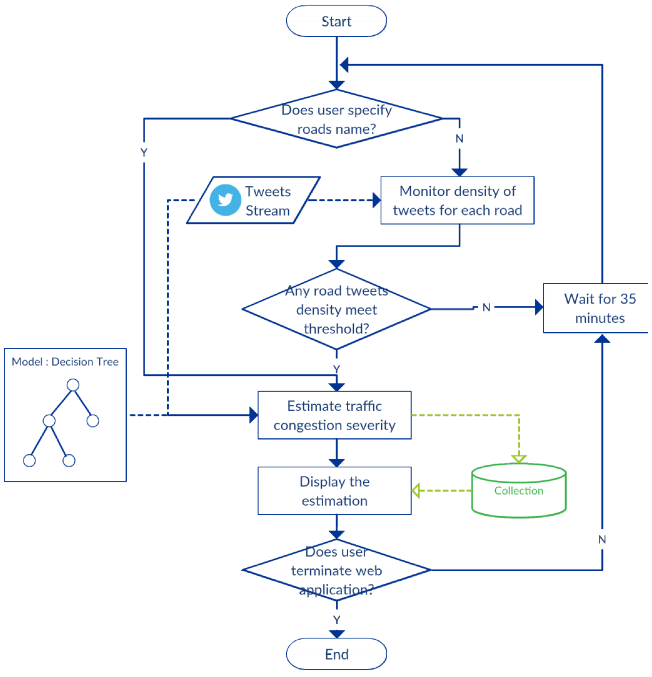

Figure 5. Use of model in web application.

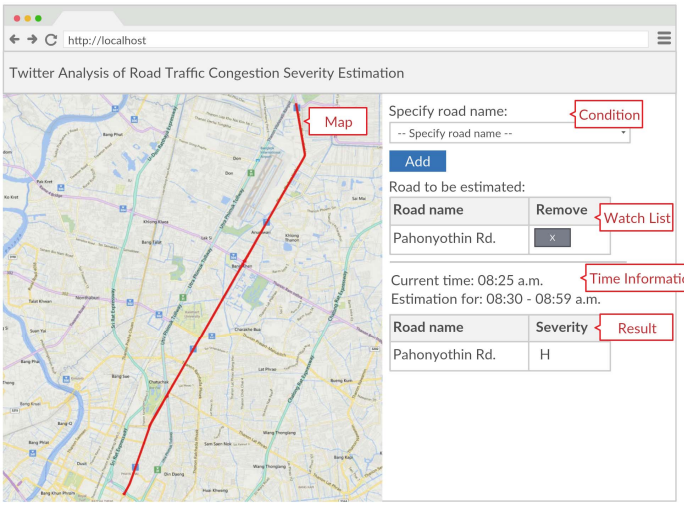

Figure 6. User interface of web application.

In addition, a user can specify a road name in the *Condition* section, and the road name will be added to the *Watch List* section. (The user can also remove it from the watch list.) The *Time Information* section displays the current time and tells the user which time duration the congestion severity estimation is for. The *Result* section displays the estimation result.

IV. Experiment and Evaluation

Since we aim to study the usefulness of Twitter messages to congestion severity prediction, in the experiment we employed several attributes sets to the C4.5 decision tree in order to test which set could give tweets density a high information gain value. The result was the attributes set shown earlier in Table I. The model from the training is shown in Fig. 7.

```
tweet_density <= 45
|   hour = H3: L (10.0)
|   hour = H5: L (10.0/2.0)
|   hour = H2: L (14.0)
|   hour = H1: L (14.0)
|   hour = H16: H (1.0)
|   hour = H15: H (4.0)
|   hour = H14: H (3.0)
|   hour = H11: H (5.0)
|   hour = H10: H (8.0)
|   hour = H8: M (2.0)
|   hour = H6: H (13.0/7.0)
|   hour = H23: L (10.0)
|   hour = H22: L (7.0)
|   hour = H21: H (1.0)
|   hour = H20: L (0.0)
|   hour = H18: L (0.0)
|   hour = H17: L (0.0)
|   hour = H7
|   |   tweet_density <= 32: L (3.0/1.0)
|   |   tweet_density > 32: H (2.0/1.0)
|   hour = H12: H (2.0)
|   hour = H9: H (10.0)
|   hour = H0: L (14.0)
|   hour = H4: L (9.0)
|   hour = H13: H (1.0)
|   hour = H19: L (0.0)
tweet_density > 45: H (180.0/11.0)
```

Figure 7. Decision tree model obtained from Weka.

The training result not only shows that tweets density has the highest information gain value and is the best attribute to classify the data, but also shows that day of week and minutes of hour are not good enough to classify the data.

We evaluated the model by 10-fold cross validation using Weka, and Table II shows the performance of the model. The attributes we employed to the C4.5 decision tree result in high precision and recall values for the class L and H, but low precision and recall for the class M. We inspected the training data and found that that the class M is imbalanced, but the weighted average performance is still high.

V. Discussion

Several issues were investigated during the construction of the model. In Section III.B, we used the weights of 10, 4, and 1 for the tweets in the *Follow*, *Keyword*, and *Geo-tagged* collections. In fact, we also experimented with other sets of weight values but selected the aforementioned since the resulting model could predict all severity classes H, M, and L. Apart from the C 4.5. decision tree, we also tried other prediction models such as decision table and Naïve Bayes, but the resulting models could not predict the severity class M. As we trained the model using one road and the training dataset contained the imbalanced class M, we further experimented if using more data of other road could help make the dataset more balanced and more attributes become useful to the model. First, we tried to add the data of another road (i.e. Ratchadapisek Road), processed them to obtain a .csv file, and merged this file with that of the Pahonyothin Road to test if the training result could be generic for these two roads. We found that the model has much lower performance than the model based on one road. In addition, we tested with the training data of the Ratchadapisek Road only, and found that using the median to represent the severity class of the whole road made the class L dominate the others. From these experiments, it could be that each road might have its own characteristics, and hence to estimate congestion severity of other roads, we should find a suitable attributes set and method to select the representative severity class for the whole road. Another limitation of using Twitter is that we generally cannot extract fine-grained information, i.e. which link of the road a particular tweet is associated with, except for a geo-tagged tweet. But a geo-tagged tweet itself may not have congestion-related content. Also, it is found that not a great number of tweets from the selected traffic accounts indicate the links of the roads. Therefore the model and the web application in the current form can only estimate congestion severity of the whole roads.

TABLE II. Performance Result

Class	Precision	Recall	F-Measure	ROC Area
L	0.904	0.825	0.863	0.893
H	0.907	0.962	0.934	0.902
M	0.25	0.125	0.167	0.695
Weighted Avg.	0.89	0.898	0.892	0.894

VI. Conclusion

This paper has shown that tweets alone can be used to estimate road traffic congestion condition as tweets density and hours of day are useful attributes for building a congestion severity prediction model. In an evaluation, the model shows good performance, and the web application can help with planning a travel by estimating congestion condition 30 minutes in advance. We plan to improve the model by trying to extract more information that can indicate road links from non-geo-tagged tweets and see if that can help better balance the dataset. Construction of a more general model, using more road data, should be further explored. In addition, the traffic-condition-related keywords can be refined to better collect tweets that are related to traffic congestion.

Acknowledgment

We would like to thank The Intelligent Traffic Information Center Foundation or iTIC (http://www.iticfoundation.org/) and Longdo Traffic website (http://traffic.longdo.com/) for providing us the actual road traffic congestion condition.

References

[1] Twitter, https://twitter.com/. Last accessed: 15 April 2016.

[2] L. Burks, M. Miller, and R. Zadeh, "Rapid estimate of ground shaking intensity by combining simple earthquake characteristics with tweets," Proc. 10th National Conf. Earthquake Engineering, 2014.

[3] F. Lécué, R. Tucker, V. Bicer, P. Tommasi, S. Tallevi-Diotallevi, and M. Sbodio, "Predicting severity of road traffic congestion using semantic web technologies," The Semantic Web: Trends and Challenges, LNCS 8465, 2014, pp. 611-627.

[4] N. Wanichayapong, W. Pruthipunyaskul, W. Pattara-Atikom, and P. Chaovalit, "Social-based traffic information extraction and classification," Proc. 11th Int. Conf. ITS Telecommunications (ITST), 2011, pp. 107-112.

[5] N. Klakhaeng, J. Yaothanee, S. Sinthupinyo, and W. Pattara-Atikom, "Traffic prediction models for Bangkok traffic data," Proc. 8th Int. Conf. Electrical Engineering / Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2011, pp. 484-487.

[6] Geofabrik GmbH, OpenStreetMap Contributors, Thailand OpenStreetMap data, http://download.geofabrik.de/asia/thailand.html. Last accessed: 15 April 2016.

[7] QGIS Development Team, QGIS, http://www.qgis.org/en/site/. Last accessed: 15 April 2016.

[8] IETF Geographic JSON Working Group, GeoJSON, http://geojson.org/. Last accessed: 15 April 2016.

[9] MongoDB, Inc., MongoDB, https://www.mongodb.org/. Last accessed: 15 April 2016.

[10] Twitter Inc., The Streaming APIs, https://dev.twitter.com/streaming/overview. Last accessed: 15 April 2016.

[11] Lexto, Thai Lexeme Tokenizer by Sansarn, http://www.sansarn.com/lexto/. Last accessed: 15 April 2016.

[12] Weka 3, http://www.cs.waikato.ac.nz/ml/weka/. Last accessed: 15 April 2016.

[13] Django Software Foundation and Individual Contributors, Django: The Web Framework for Perfectionists with Deadlines, https://www.djangoproject.com/. Last accessed: 15 April 2016.

[14] P. "fracpete" Reutemann, Python-Weka-Wrapper, http://pythonhosted.org/python-weka-wrapper/. Last accessed: 15 April 2016