RISECURE: Metro Transit Disruptions Detection Using Social Media Mining And Graph Convolution

Omer Zulfiqar, Yi-Chun Chang, Po-Han Chen, Kaiqun Fu, Chang-Tien Lu, David Solnick, and Yanlin Li

Abstract In recent years we have seen an increase in the number of public transit service disruptions due to aging infrastructure, system failures and the regular need for maintenance. With the fleeting growth in the usage of these transit networks there has been an increase in the need for the timely detection of such disruptions. Any types of disruptions in these transit networks can lead to delays which can have major implications on the daily passengers. Most current disruption detection systems did not operate in real-time or lack transit network coverage. The theme of this thesis was to leverage Twitter data to help in earlier detection of service disruptions. This work involves developing a pure Data Mining approach and an approach that uses Graph Neural Networks to identify transit disruption related information in Tweets from a live Twitter stream related to the Washington Metropolitan Area Transit Authority (WMATA) metro system. After developing the two different models, a Dynamic Query Expansion model and a Tweet-GCN to represent the data corpus we performed experiments and comparisons to other existing models, using two different benchmark datasets, to justify the efficacy of our models. After seeing the results across both the Dynamic Query Expansion and the Tweet-GCN, with an average accuracy of approximately 78.9% and 87.3% we were able to conclude that the graph neural model is superior for identifying transit disruptions in a Twitter stream and also outperforms other existing models.

Keywords Data mining · Graph convolution · Dynamic query expansion · Web application · Twitter

These authors contributed equally to this work.

O. Zulfiqar · Y.-C. Chang · P.-H. Chen · K. Fu · C.-T. Lu (✉)
Department of Computer Science, Virginia Tech, Northern Virginia Center, Falls Church, VA, USA
e-mail: omer95@vt.edu; bensonchang@vt.edu; pohan@vt.edu; fukaiqun@vt.edu; ctlu@vt.edu

D. Solnick · Y. Li
Washington Metropolitan Area Transit Authority, Washington, DC, USA
e-mail: dsolnick@wmata.com; yli@wmata.com

T. Özyer (eds.), *Social Media Analysis for Event Detection*, Lecture Notes in Social Networks, https://doi.org/10.1007/978-3-031-08242-9_5

1 Introduction

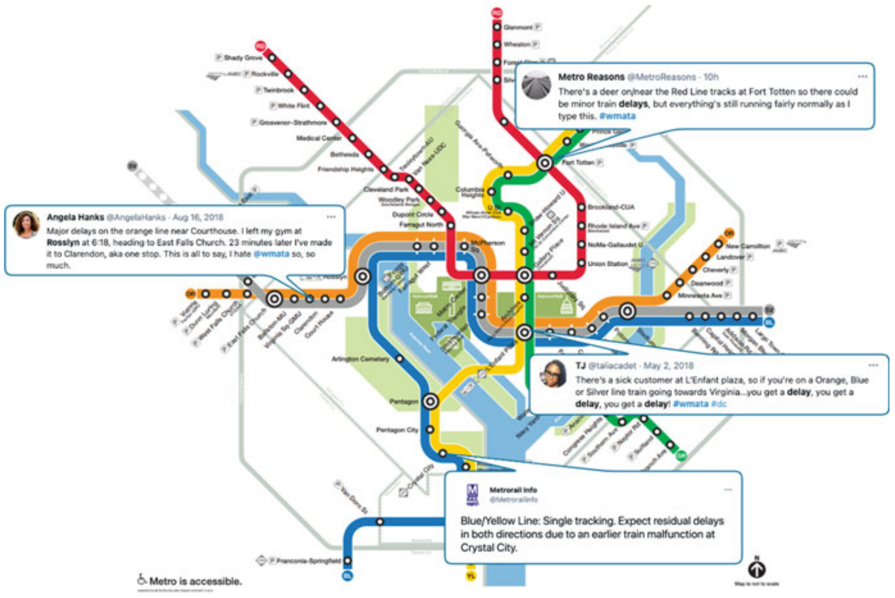
Public Transit Networks are an integral part of the infrastructure for all major metropolitan cities. Since they are virtually open to everyone, these transit systems bring in large volumes of daily users or customers. The metro/subway of any city plays an important role by connecting the suburbs and outskirts of the city to the main metropolitan area. This makes them one of the popular modes of transportation for daily commuters. [1]. Back in 2019 the Washington DC Metropolitan Transit Authority reported of having an average daily rail ridership of around 630,000 [2]. That is approximately 315,000 daily riders on the Metro on a given weekday, assuming each rider makes a round trip. Disruptions in service can severely affect these daily commuters and often force them to seek alternative modes of transportation. This could eventually drive customers away, denting the revenue generation for the transit agency.

In today's era of technological advancements, the growing use of social media applications and platforms allows the users to act as live human sensors. Anyone can post and report details of events they witness or experience outside in the physical world [3]. In 2020 it was discovered that almost 500 million Tweets are posted daily. This extensive daily use, speed and coverage of Twitter makes it a major social media platform and constantly a major source of data from which topical information on various events can be extracted. These events are represented by three main dimensions:

1. Time
2. Location
3. Entity-related information about the event and its participants.

We can extract all this information from the Twitter data and use it to our advantage. Figure 1 shows a sample of the information Twitter data contains and how Tweeters can act as surrogates or human sensors.

In our previous papers RISECURE: Metro Incidents And Threat Detection Using Social Media [1], we presented a tool that leverages Social Media Data and uses Tweets as surrogates to extract information relevant to any possible security events/incidents within a Metro system. Since our project started to move towards a collective disruption identification system we needed to improve our event extraction technique by using a more sophisticated model. In this paper we develop a Graph Neural Network based approach for text classification and event extraction. The graph neural network learns features by capturing information from it's neighbors [4]. Our proposed model involves building a single diverse text graph for the whole training corpus. The graph is developed by using the corpus from a pre-existing disruptions data set. This graph contains both word nodes and Tweet nodes allowing us to explicitly model the global word co-occurrence. After the graph is built it is fed into a convolutional neural network architecture which follows an approach similar to the work of Kipf et al.[4]. This architecture allows the


Metro Reasons @MetroReasons · 10h

There's a deer on/near the Red Line tracks at Fort Totten so there could be minor train **delays**, but everything's still running fairly normally as I type this. #wmata

Angela Hanks @AngelaHanks · Aug 16, 2018

Major delays on the orange line near Courthouse. I left my gym at **Rosslyn** at 6:18, heading to East Falls Church. 23 minutes later I've made it to Clarendon, aka one stop. This is all to say, I hate @wmata so, so much.

TJ @taliacadet · May 2, 2018

There's a sick customer at L'Enfant plaza, so if you're on a Orange, Blue or Silver line train going towards Virginia...you get a **delay**, you get a **delay**, you get a **delay**! #wmata #dc

Metrorail Info @MetrorailInfo

Blue/Yellow Line: Single tracking. Expect residual delays in both directions due to an earlier train malfunction at Crystal City.

Fig. 1 An example of how Tweeters can be used as human sensors and the disruption related information contained within Twitter data

model to scale linearly in the number of graph edges by learning the hidden layer representations that encode both local graph structure and features of nodes [4].

We will provide an overview of both Dynamic Query expansion and GCN approaches in this paper. We use the same Geo-tagging technique as in our previous paper [1]. Before we integrate a new model into our system we compare the approach with other existing text classification techniques to perform a benchmark evaluation. The major contributions of this paper are:

- **Social Media Mining:** Acquisition of Twitter data to store data on the events and use it to extract candidate Tweets using keywords detection, and using Dynamic Query Expansion to track any new and emerging chatter on the incident via Dynamic Query Expansion.
- **Graph Convolution:** We Develop and implement a GCN model to classify disruptions from Twitter data and their comparative analysis. This involves building a text graph to learn feature information from the available corpus. The proposed approach is discussed after the overview of our current system, and compared to other forms of text classification.
- **Web and Mobile Platform Generation:** A convenient provision of the Data Mining model providing users with an effective visualization of the location of the event along with any necessary information in the form of a timeline.

2 Related Work

There has been a lot of work done in the field of text classification and event extraction from social media data.

Similar to our work, Ji et al. [5] approach the disruption detection problem in transit service using Twitter data. However, they utilized a multi-task learning framework in their approach. They developed a supervised model which utilizes unique metro specific assumptions in a feature space, reflected in the two kinds of regularizers proposed in the model. They proposed an algorithm based on the ADMM framework which divides the problem into a set of sub-problems which are solved using block coordinate descent and proximal operators.

Gu et al. [6] developed a technique to mine Tweets to extract traffic incident information on highways and arterials. The developed a dictionary of important keywords and used combinations of those keywords to detect traffic incident information. Tweets were mapped to a binary vector in a feature space formed by the dictionary and labeled as incident related or not. If they were labeled as traffic incident related, they were the geo-coded and further classified into the respective incident classes. Zhang et al. [7] assessed the use of Tweets for traffic incident awareness. They developed a Latent Dirichlet Allocation (LDA) model and document clustering technique to model incident-level semantic information and also applied spatial point analysis to explore certain spatial patterns.

Traditional forms of text classification use various feature engineering techniques. Several techniques build text representations after learning word embeddings [8–10]. With development of neural networks people have used Convolution Neural Networks for sentence classification [11] and Recurrent Neural Networks for text classification using multi task learning frameworks [12]. There have been several studies where various researchers tried to develop a more general architecture similar to a CNN model that could work on arbitrary graphs. One such implementation is presented by Kipf et al. [4], who were able to use a GCN to outperform other techniques in several tasks including text classification, machine translation etc.

3 System Overview

In this section, we illustrate the system architecture of the RISECURE application, as pictured in Fig. 2. The GCN model integrated application follows a similar architecture, where instead of the query expansion module we integrate our GCN.

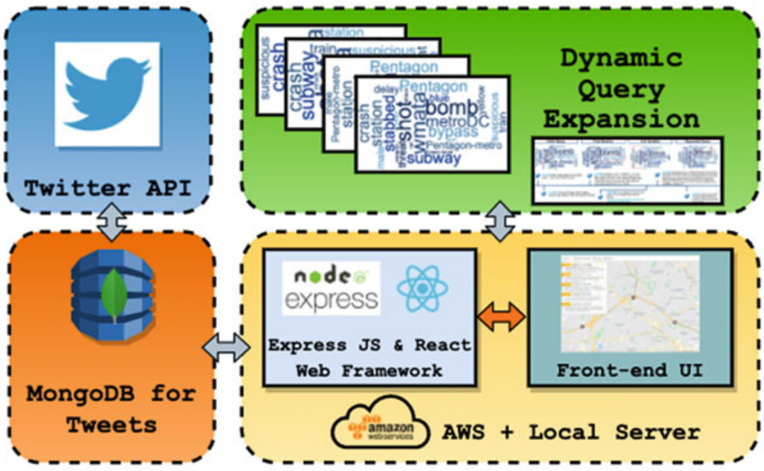


Fig. 2 System architecture

3.1 Data Acquisition

To avoid the limitation of Twitter's API, the tool GetOldTweets3 was used to collect the historic data samples. Since this study is dealing with real-time events, we will be using the Twitter Streaming API through the Python library called *Tweepy* for the system. Tweepy gives us access to real-time public Tweets by setting up query parameters based on our needs. For each city or metropolitan area there are different slogans, catchphrases, hashtags and influential users. Figure 1 previously shows a sample of such Tweets. We can use these local parameters to our advantage to help us capture the information we need. To make sure to also acquire a generalized stream of Tweets related only to WMATA, a list of specific WMATA related keywords were passed to the GetOldTweets API and Tweepy streamer. This list mostly includes the name of the stations and station localities along with their abbreviations to accommodate for variant Tweeting styles. This helps to refine the incoming data by removing unnecessary noise from the stream. Figure 3 shows a word cloud of the initial query used to acquire WMATA related Tweets.

The acquired data is first stored in AWS DynamoDB. Since we will be dealing with real-time data, DynamoDB is ideal for this study. A DynamoDB stream can be setup, which makes change data capture from the database available on an event stream. This can also be combined with AWS Elasticsearch to index our data and to perform any real-time analytics on the data, if necessary [13]. A lambda function can be incorporated as a micro-service to execute the data pre-processing script every time new data is acquired. Figure 4 shows the entire data pipeline.

Once the data pre-processing script is triggered the Tweet(s) will then be cleaned and pre-processed. Tweets can contain emotions, hashtags or special characters which makes them complex sentences. So before we pass them to our model(s)

Fig. 3 Initial WMATA Twitter API Query

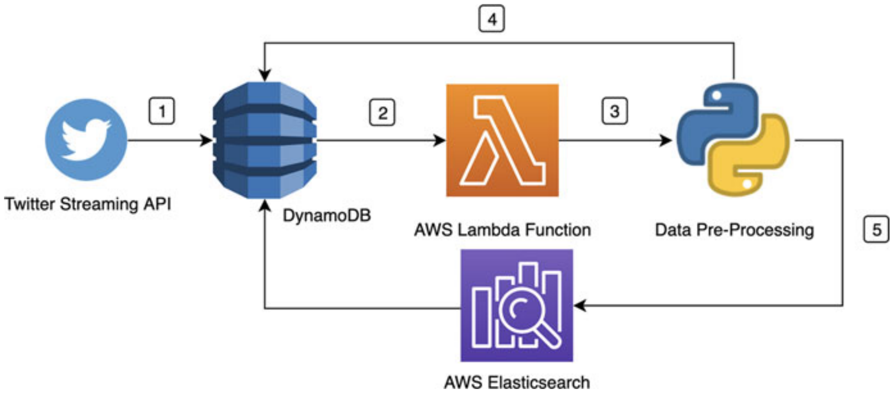


Fig. 4 Data Pipeline

we have to use textual mining techniques to clean them up. This can be done by using the Natural Language Toolkit library in python to remove non-alphanumeric characters and stop words and tokenize the Tweets. The python NLP library called *SpaCy* is also utilized for this pre-processing step(s).

For the Dynamic Query Expansion model the simple tokenized list set of Tweets is passed to the algorithm. For the GCN the Tweets were annotated before training. We used the WMATA daily service reports as the ground truth for identifying and labelling disruptions related Twitter data. We had a total of 7 different disruption classes. These were operational, mechanical, track, security, environmental, medical and non-disruption.

3.2 *Application Server*

This is the core server component of the application. We use AWS and MongoDB to help integrate the data acquisition module and the backend database. After the data has been acquired, the AWS Lambda function will trigger and send an API request to our backend service to update the data in our database. For the backend, we use Express.js with node.js as a web server framework following REST API principles.

3.2.1 Application and Mobile Interface

This is the major component of user interactions and operations. The web application was built based on the React.js framework and Google Map API. Besides, we use Progressive web application(PWA) to construct our mobile app. PWA can be installed on the user's device much like native apps and provide cross-platform compatibility for iOS and Android. The disruption incidents are accessible from the UI through 3 major components: the real time panel, the station marker and the alert notification pop ups.

3.2.2 Real Time Incidents Panel

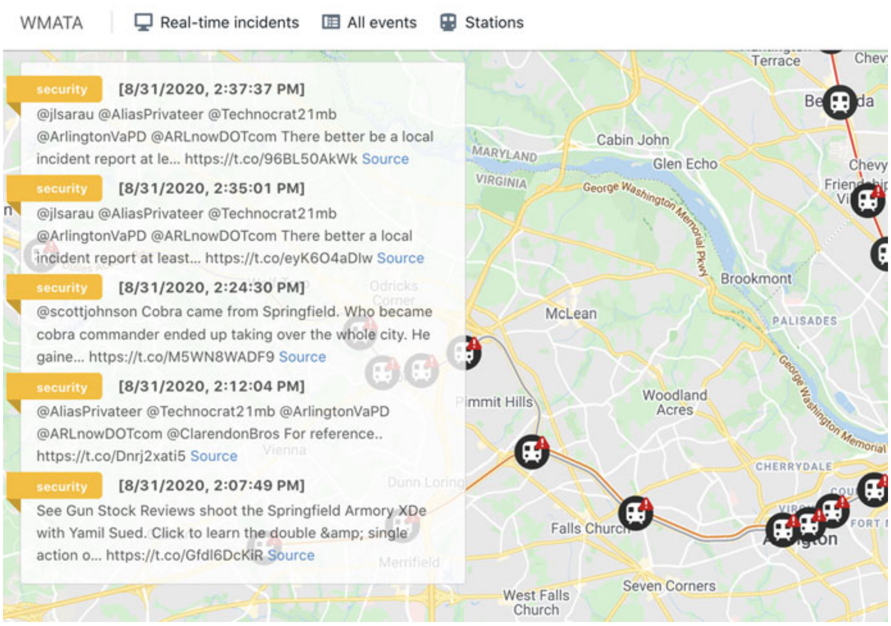
The real time panel provides the user with the latest information about any occurring incidents at any station. Tweets related to incidents are collected by timestamp and are used to construct a real-time storyline. Each incident related Tweet is tagged under a specific category which is displayed on the yellow label. The user is also provided a link to the original Tweet itself. Figure 5 provides a concept of the real-time panel.

3.2.3 Alert Notification System

The alert notification system allows users to subscribe to multiple stations and the system provides an immediate alert notification when an incident is detected. The alert notification system also updates the latest follow-up information once the authority validates the authenticity of the event. Figure 6 illustrates the scenario of our application pushing an alert notification for first event-related Tweets posted and then the verified event notification.

3.2.4 Station Marker

Station markers with a red warning sign indicate a disruption at the station due to a security incident. Clicking on the station marker will display two small pop ups.



security [8/31/2020, 2:37:37 PM]
@jlsarau @AliasPrivateer @Technocrat21mb @ArlingtonVaPD @ARLnowDOTcom There better be a local incident report at le... https://t.co/96BL50AkWk Source

security [8/31/2020, 2:35:01 PM]
@jlsarau @AliasPrivateer @Technocrat21mb @ArlingtonVaPD @ARLnowDOTcom There better a local incident report at least... https://t.co/eyK6O4aDlw Source

security [8/31/2020, 2:24:30 PM]
@scottjohnson Cobra came from Springfield. Who became cobra commander ended up taking over the whole city. He gaine... https://t.co/M5WN8WADF9 Source

security [8/31/2020, 2:12:04 PM]
@AliasPrivateer @Technocrat21mb @ArlingtonVaPD @ARLnowDOTcom @ClarendonBros For reference.. https://t.co/Dnrj2xati5 Source

security [8/31/2020, 2:07:49 PM]
See Gun Stock Reviews shoot the Springfield Armory XDe with Yamil Sued. Click to learn the double &amp; single action o... https://t.co/GfdI6DcKiR Source

Fig. 5 Real Time Incidents Panel

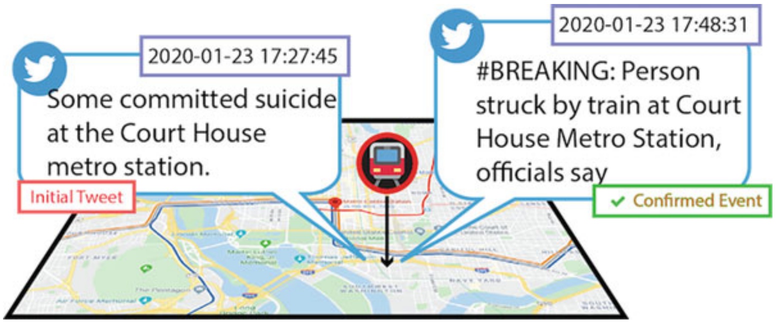
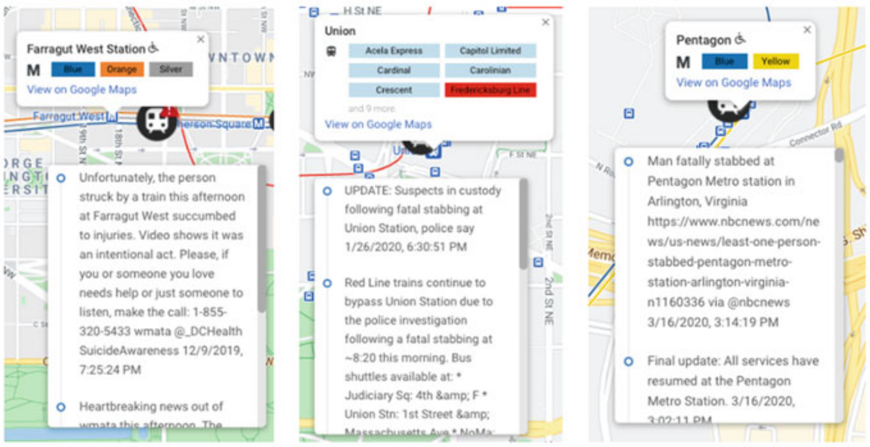

2020-01-23 17:27:45
Some committed suicide at the Court House metro station.
Initial Tweet
2020-01-23 17:48:31
#BREAKING: Person struck by train at Court House Metro Station, officials say
Confirmed Event

Fig. 6 Alert notification system

One shows the details of the station itself and the other displays a timestamped storyline which contains events specific to that particular station. This allows users to navigate to the station of their choice on the map and stay updated with any recent incidents at that station. Figure 7 shows a concept of this component.


Farragut West Station

M Blue Orange Silver

View on Google Maps

Unfortunately, the person struck by a train this afternoon at Farragut West succumbed to injuries. Video shows it was an intentional act. Please, if you or someone you love needs help or just someone to listen, make the call: 1-855-320-5433 wmata @_DCHealth SuicideAwareness 12/9/2019, 7:25:24 PM

Heartbreaking news out of wmata this afternoon. The

Union

Acela Express Capitol Limited
Cardinal Carolinian
Crescent Fredericksburg Line

and 9 more

View on Google Maps

UPDATE: Suspects in custody following fatal stabbing at Union Station, police say 1/26/2020, 6:30:51 PM

Red Line trains continue to bypass Union Station due to the police investigation following a fatal stabbing at ~8:20 this morning. Bus shuttles available at: • Judiciary Sq: 4th &amp; F • Union Stn: 1st Street &amp; Massachusetts Ave • NoMa:

Pentagon

M Blue Yellow

View on Google Maps

Man fatally stabbed at Pentagon Metro station in Arlington, Virginia https://www.nbcnews.com/news/us-news/least-one-person-stabbed-pentagon-metro-station-arlington-virginia-n1160336 via @nbcnews 3/16/2020, 3:14:19 PM

Final update: All services have resumed at the Pentagon Metro Station. 3/16/2020, 3:02:11 PM

Fig. 7 Station event list

4 Methodology

This section will go over the details of the two proposed models for this study.

4.1 Dynamic Query Expansion

After the Twitter stream returns a pool of WMATA related Tweets, a second query of keywords is used to filter out Tweets that maybe related to disruption. This query consists of keywords that were found to be able to identify incidents that may cause disruptions in the service or jeopardize the safety of the commuters and the infrastructure of the system. Table 1 shows the keywords used to identify these Tweets. This query then returns Tweets similar to the ones shown in Fig. 1. Once a disruption related Tweet is found, the dynamic query algorithm is run on that Tweet to track incident and retrieve updates.

Table 1 Keywords used for disruption query

Disruption keywords
Police
Malfunction
Slowdown
Delay
Brake
Fire
Emergency
Bypass
Single track
Uncoupled
Injury
Crash
Struck
Investigation
Disabled
Power outage
Operational
Door
Signal
Rush hour

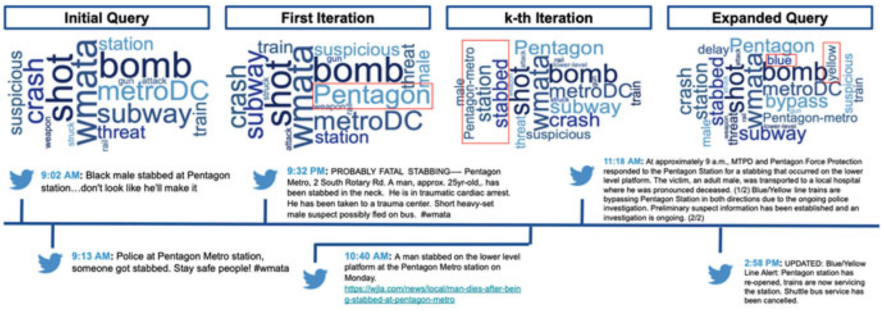
Dynamic Query Expansion evaluates inputs and reformulates the query result to improve retrieval performance. After acquiring the candidate Tweets, we can use data to extract the representative keywords for a specific threat event. This helps keep track of the emerging information for the event as it progresses. Besides, we select some high-frequency keywords, as shown in Table 1, as our initial seed query S based on analyzing the historical data of threat-related Tweets.

Algorithm 1: Dynamic Query Expansion Algorithm

```
Input: A time-ordered sequence of Tweets $\langle T_0, T_1, \ldots, T_t \rangle$, Seed Query S
Output: Expanded Query Q
Set $Q_0 = S = F_0, w(F_0) = 1, k = 0$
repeat
    $k = k + 1;$
    $w(F_k) = idf(F_k) \cdot C \cdot w(T_{(k-1)})$
    $w(T_k) = \Phi \cdot C' \cdot w(F_k));$
    repeat
        $swap(min(w(T_k)), MAX(w(T - T_k)));$
        $\sigma = min(w(T_k)) - MAX(w(T - T_k));$
    until $\sigma \leq 0;$
until $w(F_k) = w(F_{(k-1)});$
$Q = F_k;$
```

To select the representative keywords, we use the algorithm based on Dynamic Query Expansion (DQE) techniques [14, 15]. Given a time-ordered sequence of Tweets $\langle T_0, T_1, \ldots, T_t \rangle$ and Seed Query S, we could retrieve the new expanded query Q to represent this event. F_k is the feature node. W is the set of weights for nodes where higher weights denote a higher degree of relation between the node (either a Tweet or a feature) and threat-related theme. We can calculate the weight of F_k by Inverse Document Frequency(IDF) and weight of $T_{(k-1)}$. C is the adjacency matrix.

For each iteration, dynamic query expansion compares the minimum weight of the related Tweet node and the maximum weight of unrelated Tweet node, selecting the one with a higher score and putting it in the result. After the kth iteration, it converges to the stable representative keywords. After the stable status is reached, we can assume that the highest weighted keywords could describe the event. We retrieve this result and represent it on our application. The dynamic query expansion for the Pentagon Metro Stabbing Case study is shown in Fig. 8. After more event-related Tweets are collected, we can see how keywords transform from an initial query with equal weight to the expanded query with more representative keywords. The algorithm detected an initial tweet of an African American male being stabbed at Pentagon Station around 9 AM using the initial query. Over the course of the next few hours as new information comes in, the query expansion is at work. We see the word Pentagon add to the expanded query after the initial tweet, helping us identify the location of the incident. As more and more data comes in we also collect information about the disruptions caused by the incident. We see the algorithm collect information about delays on the blue and yellow lines due to an ongoing police investigation.



9:02 AM: Black male stabbed at Pentagon station…don't look like he'll make it

9:13 AM: Police at Pentagon Metro station, someone got stabbed. Stay safe people! #wmata

9:32 PM: PROBABLY FATAL STABBING— Pentagon Metro, 2 South Rotary Rd. A man, approx. 25yr-old,. has been stabbed in the neck. He is in traumatic cardiac arrest. He has been taken to a trauma center. Short heavy-set male suspect possibly fled on bus. #wmata

10:40 AM: A man stabbed on the lower level platform at the Pentagon Metro station on Monday. https://wjla.com/news/local/man-dies-after-being-stabbed-at-pentagon-metro

11:18 AM: At approximately 9 a.m., MTPD and Pentagon Force Protection responded to the Pentagon Station for a stabbing that occured on the lower level platform. The victim, an adult male, was transported to a local hospital where he was pronounced deceased. (1/2) Blue/Yellow line trains are bypassing Pentagon Station in both directions due to the ongoing police investigation. Preliminary suspect information has been established and an investigation is ongoing. (2/2)

2:58 PM: UPDATED: Blue/Yellow Line Alert: Pentagon station has re-opened, trains are now servicing the station. Shuttle bus service has been cancelled.

Fig. 8 Dynamic query expansion for pentagon metro stabbing case study

4.2 Graph Convolution

Graph Neural Networks have been commonly used for classification techniques lately. A GNN is a model that is built on the concepts of Graph Theory. A graph is a type of data structure that allows one to easily represent the relationships between different types of data it contains through nodes and edges. In the graph, a data point is represented as a node and edges connect or link multiple data points. The weights of the edges and distances between nodes define the relationship amongst the data. This text classification experiment that we are performing turns into a node classification for our proposed approach.

In text classification the nodes represent individual words or documents and the edges represent the word co-occurrence in the document or corpus depending on the type of edge. The graph is translated into a feature network and is represented as an Adjacency Matrix. For a Graph Convolutional Network the convolution operation is similar to that of a regular Convolutional Neural Network where the model learns the features by inspecting neighboring nodes in the graph. The GCN will take a weighted average of neighbor node features, including itself. The resulting feature vector is then passed through a neural network for training which learns the relationship amongst the data for classification. This neural network then returns a final vector containing the result of the classification. Rather than just identifying keywords like the dynamic query expansion technique, the GCN has a contextual learning ability which gives us the advantage against other techniques.

The GCN model here in (Fig. 9) is responsible for gleaning insight from the collected disruptions data and identifying disruptions in a live Twitter stream. It is setup, trained and developed in the following manner:

1. The training data is first cleaned and pre-processed to remove any unnecessary non alphabetic characters.
2. The word embedding or graph is generated from the training corpus. The graph can contain word nodes and Tweet nodes.

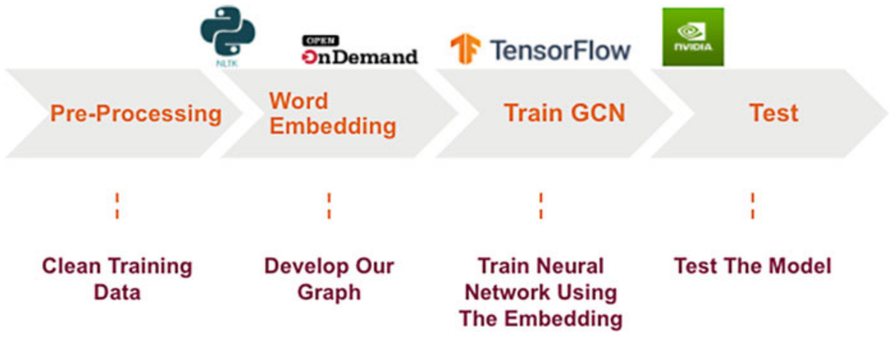


Fig. 9 GCN setup

3. This graph is then passed to a neural network which is trained by learning the relationships in the graph and then we test our model.
4. The final model is then tested by comparing it to other pre-existing Graph based text classifiers and commonly used text classification techniques. It is also tested for affect on the accuracy by tuning the parameters.

4.2.1 Building The Graph

The main component of this Tweet-GCN is the word embedding developed to train the model. A large and diversified text graph is built which contains word nodes and Tweet nodes. This allows us to explicitly model the global word co-occurrence so that graph convolution can be easily performed. In this graph, the number of nodes $|V|$ is the total number of Tweets plus the total number of unique words in the training corpus. This is just the total corpus size combined with the total vocabulary size after the preprocessing stage. A feature matrix M is set as and identity matrix ($M = I$). This means that every word and Tweet is represented as a one hot vector as the input for the Tweet-GCN. A one hot vector is just representation of categorized variables in the form of binary vectors. This helps the machine recognize categorized data much better.

The edges are built among nodes based on two properties:

1. **Tweet to Word Edges:** Based on the word occurrence or frequency in Tweets.
2. **Word to Word Edges:** Based on the word co-occurrence or frequency in the entire corpus.

The weight of an edge between a Tweet node T and word node W is the term frequency inverse Tweet frequency, which is just the TF-IDF of the word W in the Tweet T. Point-wise mutual information(PMI) was used to calculate the weights for word to word nodes. We want to quantify the likelihood of the co-occurrence of two words and PMI helps with that. PMI is a popular measure used for word associations to calculate the weight between two word nodes. Using PMI instead of only using word co-occurrence helped achieve better results in the experimentation and testing stages of the model. The weight of the edge E_{mn} between node m and node n is defined as follows:

$$E_{mn} = \begin{cases} \text{PMI}(m,n) & m, n \text{ are words, PMI}(m,n) > 0 \\ \text{TF-IDF}_{mn} & m \text{ is Tweet, } n\text{is a word} \\ 1 & m = n \\ 0 & \text{otherwise} \end{cases}$$

The PMI value for a word to word edge E_{mn} is calculated using the following equations:

$$PMI(m,n) = \log\left(\frac{p(m,n)}{p(m) * p(n)}\right) \tag{1}$$

$$p(m,n) = \frac{\#win(m,n)}{\#win} \tag{2}$$

$$p(m) = \frac{\#win(m)}{\#win} \qquad p(n) = \frac{\#win(n)}{\#win} \tag{3}$$

where $\#win(m)$ or $\#win(n)$ represents the number of sliding windows in the corpus that contain the words m and n respectively. $\#win(m,n)$ is the number of sliding windows in the corpus that contains both m and n, and $\#win$ represents the total number of sliding windows that are present in the corpus. The results from the PMI equation tell us about the semantic correlation of the two words in the corpus. A positive value indicates a high correlation, a negative value indicates little or no correlation and a value of 0 indicates that the two are statistically independent. Due to this nature, only edges with a positive PMI value were added to the graph.

4.2.2 Network Architecture

Once the graph has been built, it is fed into a multi-layer neural network architecture which follows a similar approach to Kipf et al. [4]. This architecture performs convolutions directly on the graph by inducing embedding vectors of nodes based on the properties of their neighboring nodes. The graph G can be represented using adjacency matrix A and degree matrix D. Using a single layer of convolution will allow the GCN to only capture the information from its immediate neighbors. Stacking up multiple layers gives the GCN the ability to obtain information over larger neighborhoods in the graph. For a single layer architecture, the new d dimensional node feature matrix X is computed as:

$$X_d^{(1)} = \phi(\hat{A} * X * W_0) \tag{4}$$

where $\hat{A}$ is a re-normalized adjacency matrix and W_0 is the weight matrix. ϕ is a rectifier activation function (ReLU) where $\phi(x) = max(0,x)$. For a multi-layered architecture, X is computed as:

$$X_d^{(i)} = \phi(\hat{A} * X^{(i)} * W_i) \tag{5}$$

where $X^{(0)} = X$ and i denotes the number of the current layer. For this model a two layer architecture was used. The word and Tweet node embeddings in the second layer have the same size as the labeled data set. These embeddings are then passed to a softmax classifier function:

$$y_{pred} = softmax(\hat{A} * ReLU(\hat{A}XW^{(0)})W^{(1)}) \tag{6}$$

where W_0 is an input to hidden layer weight matrix for any hidden layer with H feature maps and $W^{(0)}$ is the weight matrix for the hidden to output layer. The loss function is defined as the multi class cross entropy over the entire labeled data set.

$$L_{crossEntropy} = -\sum_{i=1}^{C_o} v_0 * \log(y_{pred}) \tag{7}$$

C_o represents the number of possible output classes, v_0 denotes one-hot encoded representation of the ground truth label and y_{pred} is the is the probability of the predicted label for the Tweet. The weight parameters were $W^{(0)}$ and $W^{(1)}$ were trained by performing batch gradient descent using the full data set for each iteration. The only downside to this approach is that it requires a large amount of memory to train the model. The two layer GCN allows for message passing between nodes that are two edges or steps apart from each other. So even though there are no predefined Tweet to Tweet edges in the graph, the two layer GCN allows pairs of Tweets to exchange information between each other.

Figure 10 shows an overview of the Tweet-GCN model. Nodes beginning with T are Tweet nodes and the rest are word nodes. Tweet to word edges are denoted by the solid black lines and word to word edges are denoted by solid red lines. $E(x)$ here denotes the embedding representation for x. For example, $E(track)$ is the embedding representation of the word track. The different colors here indicate the different classes of disruptions in the data set. Only four classes have been shown here to avoid a cluttered schematic.

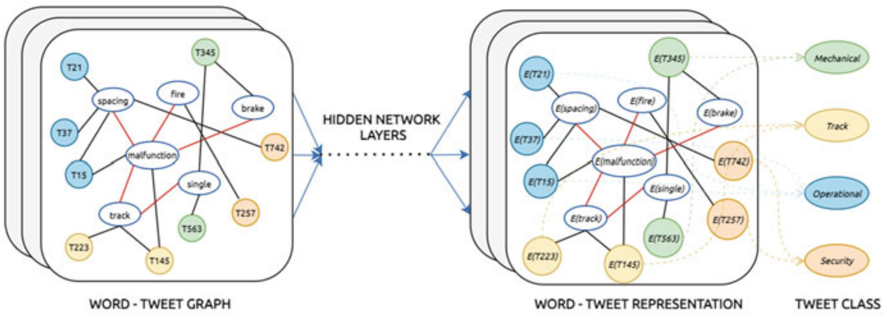


Fig. 10 Overview of Tweet-GCN

5 Experiment and Results

Our data set included 60000 disruption incidents from between 2012–2019. We also collected data from WMATA service reports which was used as the ground truth for labeling our Twitter data.

5.1 Benchmark Evaluations

We fine tuned our parameters accordingly to achieve the best results in each case, which are discussed later. For our GCN we started by setting the learning rate as 0.02,, dropout rate to 0.5, the embedding size of the convolution layer to 200 and the window size to 15. 10% of our training data was randomly selected as our validation set. We used the following baseline models:

1. **TFIDF + Regression:** Look at a basic bag of words(BOW) model with a Logistic Regression classifier.
2. **LSTM:** The LSTM model implemented by Liu et al. [12]. It uses the previous hidden state as the representation of the entire text, with and without pre-trained word embeddings.
3. **CNN:** Using the Kim et al. [11] implementation of the Convolutional Neural Network which uses max pooling on the embeddings to generate the text representations. The non-static CNN approach is used, which uses pre-trained word embeddings.
4. **Graph CNN-C:** A graph CNN model by Defferrard et al. [16] that operates convolutions over word embedding similarity graphs by utilizing a Chebyshev filter.
5. **Graph CNN-F:** A graph CNN model by Henaff et al. [17] which is similar to Graph-CNN C but instead utilizes a Fourier filter.

For the Tweet-GCN we used the optimal parameters based on the results of our parameter tests and for the baselines the default parameters were used as discussed in the papers of their original implementations along with the pre-trained 300 dimensional Stanford NLP GloVe word embedding where ever necessary. 10% of the training set was randomly selected as the validation set and following the approach of Kipf et al. [4] both GCNs were trained for a maximum of 200 epochs with early stopping enabled for no changes in validation loss for 10 consecutive epochs.

Tweet-GCN Setup A dimension size of 200 was set, a window size of 12, a learning rate of 0.02, dropout probability of 0.5 and an L2 regularization or weight decay of 0.

Four different metrics were used to evaluate the performance of the proposed approaches against the baselines. These were accuracy, precision, recall and F-score. The weighted average of all four metrics across 100 Runs for each model was reported. The accuracy measure is one of the most common metrics for evaluation. Generally, higher the accuracy is, the better the classifier is at identifying class labels. Precision gives us a measure of the relevant data points by identifying what proportion of the predicted positives is truly positive. Recall gives us a measure of how accurately the model is able to identify the relevant data points by telling us what proportion of true positives have been correctly classified. Most of the times there is a trade-off between the precision and recall scores. Sometimes these values may conflict, so they must be considered comprehensively. For this problem we would prefer to have a good value for both measures since we want to be able to identify as many disruptions as possible as precisely as possible. That is where the F-score comes in, which is the harmonic mean of the precision and recall. The F-score also does a better job at describing models dealing with class imbalance in multi-class classification problems.

From Table 2 it can be seen that all the graph based models outperform the rest of the models. This is likely due to the characteristics of the graph structure enabling the word nodes to learn the representations more accurately. Something which is impossible for the other traditional models. We see lower accuracy results and testing scores for the Dynamic Query Expansion model compared to the baselines. The Dynamic Query Expansion and keyword extraction model does not have a learning ability like the graph based models and other deep learning models therefore it is at a disadvantage when it comes to learning the semantic relationship among the data and capturing relevant disruptions data within the stream. The Tweet-GCN performs well because the graph is able to capture both Tweet-word relations and global word-word relations and because the label information of the Tweet nodes can be passed to the adjacent word nodes and relayed to other word and document nodes that are at most two steps away. This allows the Tweet label information to be propagated throughout the graph.

Graph CNN-C and Graph CNN-F use similar graph models as ours but the word nodes are connected over larger windows without weighted edges. Due to the lack of trainable edges those models are then unable to learn the significant relationships between different words. We also notice that the CNN and LSTM models provide

Table 2 Test scores for disruption classification with for the twitter disruptions data set: the results are the average of 100 runs for each model

Model	Accuracy	Precision, Recall, F-Score
Logistic Regression + TF-IDF	0.8101 ± 0.0018	0.81, 0.80, 0.81
LSTM	0.7743 ± 0.0087	0.77, 0.78, 0.78
LSTM(GloVe)	0.8237 ± 0.0163	0.81, 0.82, 0.82
CNN(GloVe)	0.7781 ± 0.0048	0.78, 0.80, 0.79
Graph CNN-C	0.8204 ± 0.0032	0.82, 0.78, 0.80
Graph CNN-F	0.8371 ± 0.0015	0.83, 0.84, 0.83
Dynamic Query Expansion	0.7892 ± 0.0153	0.75, 0.79, 0.77
Tweet-GCN	0.8726 ± 0.0021	0.87, 0.86, 0.87

satisfactory results on both datasets, but lack in their contextual information learning ability compared to the our GCN. However, both those models use pre-trained word embeddings while our GCN only uses the information provided to it by the input corpus.

5.2 *Parameter Testing*

We also performed tests by varying the parameters of our GCN.

5.2.1 Size of Sliding Window

In Fig. 11 we have the results from varying the sizes of sliding windows in the model.

We see the test accuracy be the highest for a window size of 15, and the accuracy begins to decrease when the window size becomes larger than that. Small window sizes are unable to generate enough word co-occurrence while information too large window sizes are may add extra edges to nodes that might not be closely related [4]. Since Tweets are a form of micro-blogs or short text this behavior is understandable, suggesting that a small window size may not be able to capture enough information

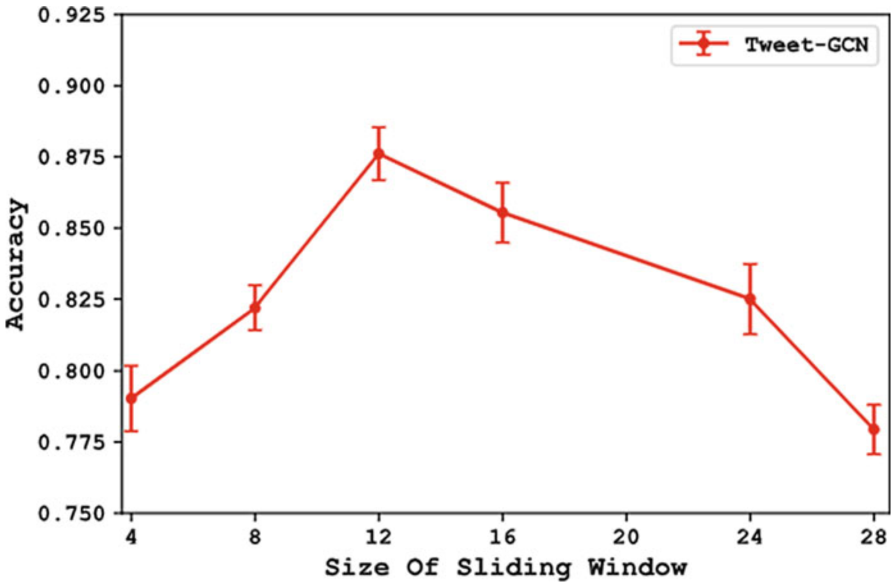


Fig. 11 Accuracy with varying window sizes

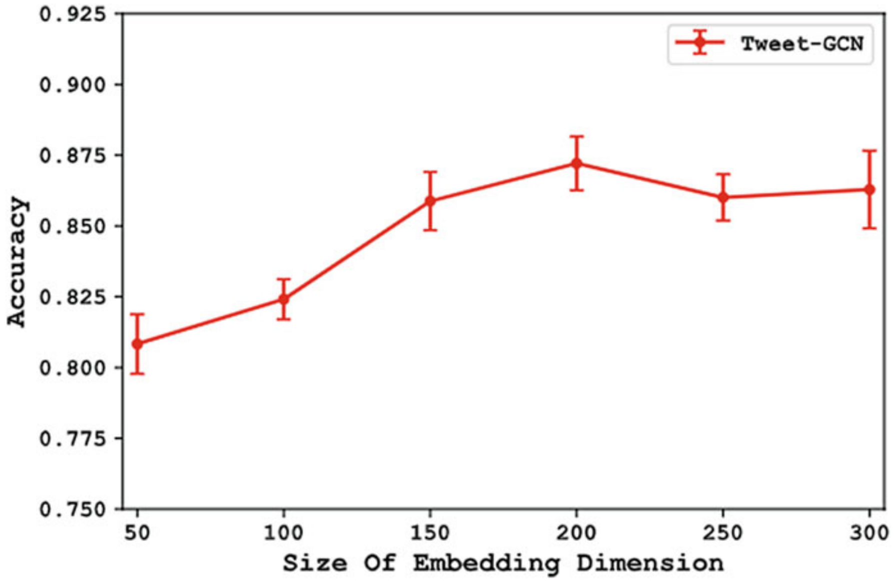


Fig. 12 Accuracy with varying embedding sizes

where as a large window size may capture extra information by adding extra edges between nodes that might not be closely related.

5.2.2 Size of Embedding Dimension

Figure 12 shows the test accuracy results with varying embedding sizes. We observed trends similar to our earlier test with the sliding windows. For small dimensions, the embeddings may not disseminate throughout the graph while large embeddings increase training time and do not change the our results by much. We found a dimension size of 200 to be optimal for both data sets.

5.2.3 Size of Training Data

To avoid a cluttered graph, we selected the best individual performing models to see how changing the size of our labeled training data effects the models. Figure 13 shows the test accuracy results of these tests on 1%, 5%, 10%, 15%, 20% and 25% of the Twitter training data set.

It can be seen that Tweet-GCN performs better by achieving higher test accuracies throughout for the partial training set. We can see the Tweet-GCN achieves an accuracy of 0.8132 ± 0.0132 for only 25% of the training data set. These results are even higher than those of some baseline models when they were trained

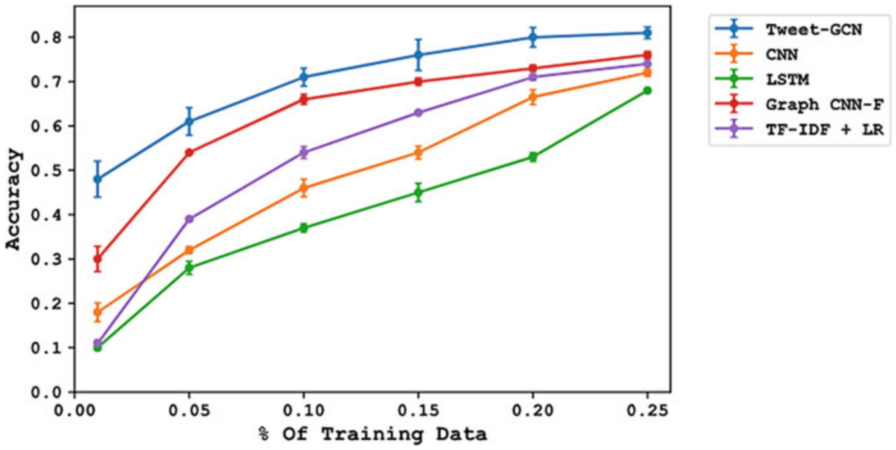


Fig. 13 Effect of size of Twitter training data on model performances

using the entire training set. These results suggest that the proposed GCN model perform reasonably well with a limited label rate and can spread and preserve label information within the graphs giving themselves the upper-hand at identifying disruption data.

From the results of our experiment we can see that our GCN model is able to achieve pretty good test results for classifying WMATA related disruptions within Twitter data. However there are still some limitations with the GCN when it comes to unlabeled information in the training data. Overcoming this hurdle will be part of some future work we look to accomplish.

6 Conclusion

RISECURE is an open-source and automated system that is capable of detecting transit disruptions by using Social Media data mining and deep learning techniques. We proposed two approaches; Dynamic Query Expansion and Tweet-GCN. The effectiveness of our proposed approaches is displayed through benchmark evaluations against other baseline models. We saw the Tweet-GCN give us the best results for identifying disruptions with an overall accuracy of 87.3%, whereas the Dynamic Query Expansion model delivered the lowest scores with an overall accuracy of 78.9%. However, the Tweet-GCN consumes a lot of memory due to the high number of edges in the corpus level graph. For future we will look to modify the model to solve this issue. For real-world deployment in transit systems such as metro rails, our proposed approach can serve as a supplementary resource to aid in swift disruption detection and, gain situational awareness. We foresee a great potential to take this platform to a higher level where it can help improve the rider experience for the public transit systems.

References

1. Zulfiqar O, Chang Y-C, Chen P-H, Fu K, Lu C-T, Solnick D, Li Y (2020) Risecure: metro incidents and threat detection using social media. In: Proceedings of the 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)
2. Metrorail ridership grew by 20,000 trips per weekday in 2019 (2020). https://www.wmata.com/about/news/2019-Metrorail-ridership.cfm
3. Zheng Y, Capra L, Wolfson O, Yang H (2014) Urban computing: concepts, methodologies, and applications. ACM Trans Intell Syst Technol 5:13838–13855
4. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. CoRR abs/1609.02907. arXiv:1609.02907
5. Ji T, Fu K, Self N, Lu C-T, Ramakrishnan N (2018) Multi-task learning for transit service disruption detection. In: Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, New York, pp. 634–641
6. Gu Y, Qian S, Chen F (2016) From twitter to detector: Real-time traffic incident detection using social media data. Transportation Research Part C: Emerging Technologies 67:321–342. https://doi.org/10.1016/j.trc.2016.02.011
7. Zhang S (2015) Using twitter to enhance traffic incident awareness. In: Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pp 2941–2946. https://doi.org/10.1109/ITSC.2015.471
8. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: International Conference on Machine Learning. PMLR, pp 1188–1196
9. Joulin A, Cissé M, Grangier D, Jégou H et al (2017) Efficient softmax approximation for gpus. In: International Conference on Machine Learning. PMLR, pp. 1302–1310
10. Tang J, Qu M, Mei Q (2015) Pte: Predictive text embedding through large-scale heterogeneous text networks. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 1165–1174
11. Kim Y (2014) Convolutional neural networks for sentence classification. CoRR abs/1408.5882. arXiv:1408.5882
12. Liu P, Qiu X, Huang X (2016) Recurrent neural network for text classification with multi-task learning. In: Twenty Fifth International Joint Conference on Artificial Intelligence
13. Gurreiro M, Megler V (2020) Detect change points in your even data stream using Amazon Kinesis Data Streams, Amazon DynamoDB and AWS Lambda. Amazon, Washington
14. Zhao L, Chen F, Dai J, Hua T, Lu C-T, Ramakrishnan N (2014) Unsupervised spatial event detection in targeted domains with applications to civil unrest modeling. PloS One, 9(10), e110206
15. Khandpur RP, Ji T, Ning Y, Zhao L, Lu C-T, Smith ER, Adams C, Ramakrishnan N (2017) Determining relative airport threats from news and social media. In: Twenty-Ninth IAAI Conference
16. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. CoRR abs/1606.09375
17. Henaff M, Bruna J, LeCun Y (2015) Deep convolutional networks on graph-structured data. CoRR abs/1506.05163