การวิเคราะห์และแสดงผลข้อมูลจากทวิตเตอร์ในเชิงภูมิศาสตร์บนกูเกิลเอิร์ธ

นางสาวอิสราภรณ์ วิทยวิรานนท์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2556

GEOGRAPHICAL ANALYSIS AND VISUALIZATION OF TWITTER DATA ON GOOGLE EARTH

Miss Isaraporn Vithyaviranont

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2013

หัวข้อวิทยานิพนธ์	การวิเคราะห์และแสดงผลข้อมูลจากทวิตเตอร์ในเชิงภูมิศาสตร์บนกูเกิลเอิร์ธ
โดย	นางสาวอิสราภรณ์ วิทยวิรานนท์
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร. วีระ เหมืองสิน

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร. บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(ศาสตราจารย์ ดร. ประภาส จงสถิตย์วัฒนา)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ผู้ช่วยศาสตราจารย์ ดร. วีระ เหมืองสิน)

.....กรรมการ

(รองศาสตราจารย์ ดร. ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

.....กรรมการภายนอกมหาวิทยาลัย

(รองศาสตราจารย์ ดร. วรเศรษฐ สุวรรณิก)

อิสราภรณ์ วิทยวิรานนท์ : การวิเคราะห์และแสดงผลข้อมูลจากทวิตเตอร์ในเชิงภูมิศาสตร์บนกูเกิลเอิร์ธ. (GEOGRAPHICAL ANALYSIS AND VISUALIZATION OF TWITTER DATA ON GOOGLE EARTH) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร. วีระ เหมืองสิน, 80 หน้า.

ทวิตเตอร์เป็นสื่อสังคมแบบไมโครบล็อกและส่งข้อความที่ได้รับความนิยมที่สุดแห่งหนึ่งด้วย Streaming API และบริการระบุพิกัดบนข้อความทำให้ทวิตเตอร์เป็นแหล่งข้อมูลขนาดใหญ่สำหรับการวิเคราะห์สื่อสังคมในเชิงพื้นที่และเวลา บทความนี้นำเสนอการออกแบบและพัฒนาระบบแอพพลิเคชั่นบนเว็บซึ่งเรียกว่า World of Tweeties (WOT) เพื่อใช้เป็นเครื่องมือในการวิเคราะห์ข้อมูลเชิงพื้นที่และเวลาของข้อความทวิตเตอร์ ผู้ใช้งานสามารถค้นหาข้อความทวีตด้วยคีย์เวิร์ด, ชื่อผู้ใช้ และช่วงเวลา รวมถึงการค้นหาด้วยหลายคีย์เวิร์ดเพื่อการเปรียบเทียบข้อมูล สำหรับการวิเคราะห์จะมีทั้งเชิงพื้นที่ซึ่งแสดงข้อมูลบนกูเกิลเอิร์ธ และเชิงเวลาจะถูกนำเสนอในรูปของกราฟจำนวนทวีตบนแกนเวลา นอกจากนี้ระบบยังแสดงผลจากการคำนวณทางสถิติและข้อมูลเชิงพื้นที่แบบตัวเลขอีกด้วย ผลลัพธ์ที่ได้สามารถจัดเก็บในคลังข้อมูลเพื่อให้เรียกดูได้ในภายหลัง การทดลองวิเคราะห์ทวีตในประเทศไทยให้ผลลัพธ์ที่น่าสนใจและมีประโยชน์ในหลายแนวทาง เช่น ในการสำรวจด้านการเมือง ติดตามสภาพการจราจร สภาพอากาศ หรือวิเคราะห์การตลาด เป็นต้น

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2556

5471038321 : MAJOR COMPUTER SCIENCE
KEYWORDS: TWITTER / GOOGLE EARTH / SPATIAL ANALYSIS / SOCIAL MEDIA ANALYSIS

ISARAPORN VITHYAVIRANONT: GEOGRAPHICAL ANALYSIS AND VISUALIZATION OF TWITTER DATA ON GOOGLE EARTH. ADVISOR: ASST. PROF. VEERA MUANGSIN, Ph.D., 80 pp.

Twitter is a popular social media system with micro-blogging and messaging features. With its Streaming API and geolocation attached data, Twitter provides a huge data source for spatial-temporal analysis on social media. This thesis presents the design and development of a Web-based application called “World of Tweeties (WOT)”. It is an on-line spatial-temporal analysis tool for Twitter data. The user can search the tweets by keyword, username, and time. Searching with multiple keywords for comparison is supported. The Google Earth plug-in provides the geographic user interface for identifying the area of interest and data visualization. Temporal analysis is supported with a time-frequency graph. The application also performs statistic and spatial analysis. The results can be archived for later examination. Experiments on the tweets originated in Thailand show interesting results and the potentials of the system in many areas such as political survey, traffic monitoring, weather monitoring, brand analysis, etc.

Department: Computer Engineering Student's Signature .....
Field of Study: Computer Science Advisor's Signature .....
Academic Year: 2013

กิตติกรรมประกาศ

ขอขอบพระคุณผู้ช่วยศาสตราจารย์ ดร.วีระ เหมืองสิน อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่สละเวลาคอยให้คำปรึกษา คำแนะนำ ข้อคิด และความช่วยเหลือต่างๆ ตลอดระยะเวลาที่ศึกษาและวิจัย จนกระทั่งวิทยานิพนธ์ลุล่วงไปด้วยดี

ขอขอบพระคุณรองศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนาประธานคณะกรมการสอบวิทยานิพนธ์ รองศาสตราจารย์ ดร.วรเศรษฐ สุวรรณิก และรองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา กรรมการสอบวิทยานิพนธ์ที่ให้ข้อชี้แนะในการปรับปรุงงานวิทยานิพนธ์ให้มีคุณภาพยิ่งขึ้น

ขอขอบพระคุณคณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้อบรม สั่งสอน ให้ความรู้ต่างๆ ซึ่งเป็นประโยชน์ต่อการทำวิจัยและการทำงานในอนาคต

ขอขอบคุณนายชัยธวัช ภวภูวดล นางสาวอรอนงค์ องค์ศิริพร และเพื่อนๆ พี่ๆ น้องๆ นิสิตสาขาวิทยาการคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ ที่คอยให้ความช่วยเหลือเกื้อหนุน แบ่งปันข้อคิดเห็นจากประสบการณ์และช่วยตักเตือนขั้นตอนต่างๆ ในการทำวิจัย

ขอขอบคุณ คุณศุภชัย คุณติสุข คุณณัฐธยาน์ กานต์สุภัคพงษ์ และเพื่อนร่วมงานที่คอยเป็นกำลังใจทั้งให้คำปรึกษาและการช่วยเหลือต่างๆ อันเป็นปัจจัยสำคัญที่ทำให้วิทยานิพนต์ฉบับนี้สามารถดำเนินการได้สำเร็จตามที่คาดหวังไว้

ท้ายที่สุดนี้ขอขอบพระคุณ คุณแม่ และครอบครัวที่คอยให้การสนับสนุนและกำลังใจหลักซึ่งช่วยผลักดันด้านการศึกษาจนประสบผลสำเร็จดังเช่นทุกวันนี้

สารบัญ

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

หน้า

หน้า

บทที่ 1
บทนำ

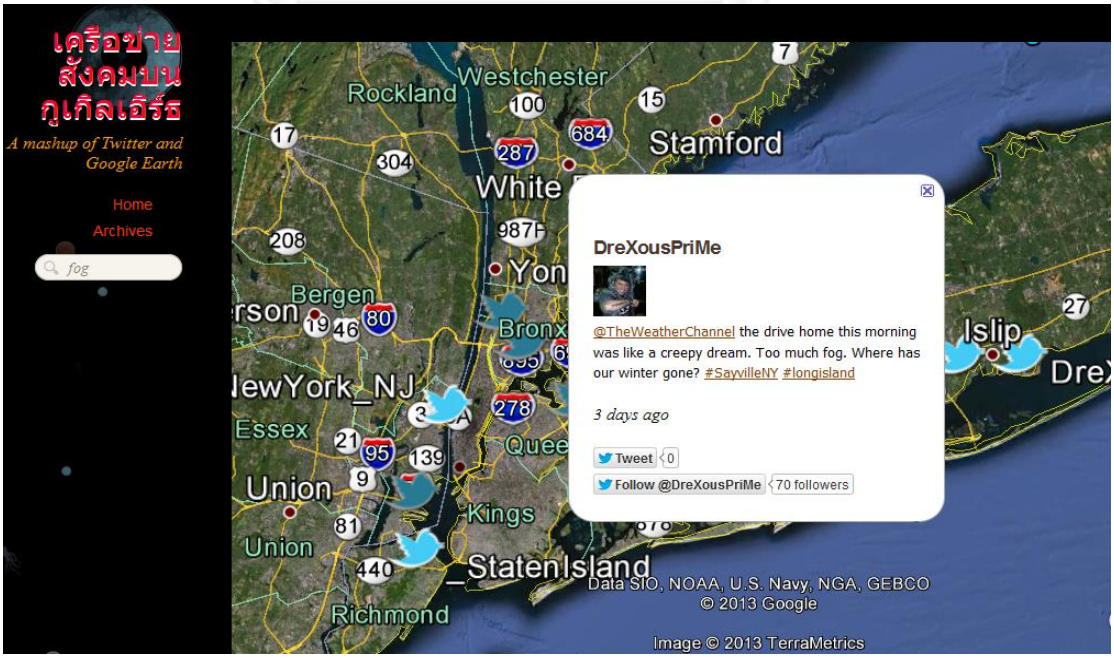
1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันสื่อสังคมออนไลน์หรือ online social media กำลังเป็นที่นิยมอย่างมากเพราะเป็นสื่อที่ใช้ต้นทุนน้อย แต่สามารถให้ผลกระทบกับคนหมู่มากได้ นอกจากนั้นยังสามารถใช้ประโยชน์ได้ตั้งแต่ส่วนบุคคล ระดับธุรกิจ ภาครัฐ ภาคเอกชนต่างๆ เพื่อการพูดคุยและกระจายข่าวสาร ด้วยความต้องการที่มากขึ้นนี้จึงไม่น่าแปลกใจนักที่ในปัจจุบันจะมีสื่อออนไลน์เพิ่มจำนวนขึ้นอย่างล้นหลาม ทั้งยังได้รับความนิยมจากผู้บริโภคอย่างทั่วถึงกัน เช่น Facebook, Twitter, Google+, Instagram และอื่นๆ แต่ละสื่อย่อมมีเอกลักษณ์เป็นของตัวเองและมีจุดมุ่งหมายที่แตกต่างกัน ดังนั้นผู้ใช้งานบางคนจึงเลือกที่จะใช้งานมากกว่าสองบริการควบคู่กันไป

ทวิตเตอร์ (Twitter) เป็นบริการเครือข่ายสังคมออนไลน์จำพวกไมโครบล็อกซึ่งถูกก่อตั้งเมื่อเดือนมีนาคมปี 2006 โดย Jack Dorsey, Evan William และ Biz Stone [1] ทวิตเตอร์อนุญาตให้ผู้ใช้งานสามารถส่งข้อความที่เรียกกันว่า ทวีต (Tweet) ได้ความยาวไม่เกิน 140 ตัวอักษร ซึ่งข้อความที่ถูกอัพเดตจะปรากฏบนหน้าเพจของผู้ใช้ ผู้ใช้คนอื่นๆ สามารถเลือกได้ว่าจะรับข้อมูลข่าวสารจากผู้ใช้คนนั้นหรือไม่ พฤติกรรมแบบนี้จะถูกเรียกว่าการติดตาม (Follow) เมื่อเลือกที่จะติดตามผู้ใช้คนอื่นข้อความของพวกเขาเหล่านั้นก็จะมาปรากฏบนหน้าเพจของผู้ใช้ด้วย ทำให้ทวิตเตอร์เป็นช่องทางรับข่าวสารที่รวดเร็วและกระชับฉับไวที่สุดทางหนึ่ง ทั้งยังไม่ยุ่งยากในการใช้งานอีกด้วย และในกรณีที่ข้อความทวีตนั้นๆ เป็นที่ถูกใจหรือเห็นดีด้วย ผู้ใช้ยังสามารถส่งต่อข้อความนั้นไปบนเพจของตัวเองเพื่อให้ผู้ติดตามของตนได้รับข่าวสารนี้เช่นกัน การส่งต่อในลักษณะนี้จะถูกเรียกว่าการรีทวีต (Retweet) ดังนั้นจะเห็นได้ว่าข้อความหนึ่งๆ สามารถถูกส่งต่อและถ่ายทอดออกไปได้อย่างไม่มีที่สิ้นสุด ตราบเท่าที่เนื้อความนั้นยังมีความน่าสนใจอยู่ ข่าวสารที่ได้รับจึงไม่ได้ปิดกั้นแค่ในวงเพื่อนหรือคนรู้จัก แต่เป็นการเปิดโลกเพื่อรับข่าวสารที่อาจข้ามทวีปมา ด้วยคุณสมบัติของทวิตเตอร์ที่มีความสามารถในการแพร่กระจายข่าวสารได้อย่างรวดเร็วและกว้างขวางจึงได้รับความนิยมเป็นอย่างมากทั่วโลก โดยพบว่าทุกวันนี้ทวิตเตอร์มีจำนวนผู้ใช้งานแต่ละวันมากกว่า 100 ล้านคน ซึ่งส่งข้อมูลไม่ต่ำกว่า 500 ล้านทวีตต่อวัน สำหรับในประเทศไทยมีผู้ใช้งานจริงมีประมาณ 200,000 คนต่อวัน ซึ่งส่งกว่า 1.1 ล้านทวีตต่อวัน [2]

เนื่องจากสื่อสังคมออนไลน์อย่างทวิตเตอร์มีข้อมูลและข้อคิดเห็นจำนวนมหาศาลไหลเวียนอยู่ในระบบ จึงมักถูกนำไปใช้เป็นเครื่องมือในการวัดกระแสสังคมด้วยหลากหลายวิธีการ ทว่าโดยมากแล้วจะเน้นการวิเคราะห์ส่วนของข้อความในทวีต เช่น การทำเหมืองข้อมูลจากกองข้อมูลมหาศาลในทวิตเตอร์ให้เป็นกลุ่มความคิดเห็นแง่ลบและแง่บวก [3] หรืออย่างการนำไปทำนายผลการเลือกตั้ง [4] การค้นหาจุดกำเนิดของการแผ่นดินไหว [5] เป็นต้น และแน่นอนว่าในประเทศไทยเองเริ่มมีการนำมาใช้ในเชิงพาณิชย์สำหรับการติดตามข่าวสาร (monitoring) แทนการนั่งอ่านข้อความด้วยมนุษย์เองแล้ว [6] ทำให้องค์กรประหยัดทั้งค่าใช้จ่ายและทรัพยากรของบริษัทไปได้มากทีเดียว

ทว่าการวิเคราะห์ข้อมูลนั้นยังอยู่ในกรอบการวิเคราะห์เชิงเวลา ซึ่งน่าเสียดายเมื่อทวิตเตอร์ยังมีคุณสมบัติอีกประการหนึ่งที่น่าสนใจ นั่นก็คือ ความสามารถในการระบุตำแหน่งทางภูมิศาสตร์ (ละติจูด ลองจิจูด) ในแต่ละข้อความซึ่งทำให้รู้ว่าถูกโพสมาจากที่ใด ข้อมูลนี้ทำให้สามารถแสดงตำแหน่งทวีตบนแผนที่ได้ ตัวอย่างเช่น โครงงานวิศวกรรม เรื่อง “เครือข่ายสังคมบนกูเกิลเอิร์ธ” ของเจตพล ติ๋วตระกูล [7] ที่นำข้อความที่ถูกโพสบนทวิตเตอร์มาปักหมุดบนแผนที่โลกในเบราว์เซอร์ ซึ่งทำให้ผู้ใช้งานแอพพลิเคชั่นนี้มองเห็นถึงการแพร่กระจายของข่าวสาร หรือความนิยมในหัวข้อต่างๆ ของผู้ใช้ทวิตเตอร์ในเชิงพื้นที่ได้เด่นชัดมากยิ่งขึ้น ดังตัวอย่างในรูปที่ 1.1 แสดงผลของการค้นคำว่า fog ผลลัพธ์แสดงให้เห็นว่าบริเวณที่มีความหนาแน่นของทวีตมากก็สะท้อนให้เห็นว่าบริเวณนั้นน่าจะมีปัญหาเรื่องหมอกลงจัด



ภาพที่ 1.1 เครือข่ายสังคมบนกูเกิลเอิร์ธ

อย่างไรก็ตาม โครงงานดังกล่าวเพียงแต่นำข้อความมาแสดงบนแผนที่เท่านั้น แม้ว่าความหนาแน่นของทวีตจะเป็นประโยชน์ในระดับหนึ่ง แต่ก็เป็นการดูด้วยสายตา ยังไม่ได้นำข้อความเหล่านั้นมาวิเคราะห์ด้วยการคำนวณที่ชัดเจน ผู้วิจัยจึงสนใจศึกษาการวิเคราะห์ชุดข้อมูลในเชิงภูมิศาสตร์ เพื่อค้นคว้าและหาวิธีการมานำเสนอถึงลักษณะการแพร่กระจายข่าวสารหรือพฤติกรรมของเครือข่ายสังคมที่แลกเปลี่ยนข้อมูลกันในเรื่องใดเรื่องหนึ่ง ซึ่งผู้วิจัยเล็งเห็นถึงคุณประโยชน์ของงานวิจัยนี้ในอนาคตในหลากหลายแนวทาง ยกตัวอย่างเช่น เมื่อมีการเปิดตัวผลิตภัณฑ์ใหม่ บริษัทเจ้าของผลิตภัณฑ์อาจจะต้องการรู้ว่าข่าวการเปิดตัวผลิตภัณฑ์นี้เป็นที่สนใจมากน้อยเพียงใด การดูจากปริมาณข้อความที่เกี่ยวข้องกับผลิตภัณฑ์นั้นในสื่อสังคมออนไลน์อาจจะตอบคำถามนี้ได้ แต่หากสามารถวิเคราะห์ในเชิงภูมิศาสตร์ที่สามารถบอกถึงภูมิภาคหรือประเทศของกลุ่มผู้ให้ความสนใจได้ด้วย ก็จะช่วยให้บริษัทสามารถวางแผนการประชาสัมพันธ์และการตลาดได้อย่างเหมาะสม หรือแม้แต่เรื่องเล็กๆ น้อยๆ อย่างสภาพการจราจรในพื้นที่หนึ่งๆ ก็สามารถกะเกณฑ์ได้ว่า ช่วงเวลาใดที่การจราจรหนาแน่นบนถนนเส้นนั้นเป็นพิเศษ ซึ่งช่วยให้ผู้ใช้ถนนสามารถวางแผนแล้วหลีกเลี่ยงเส้นทางจราจรได้ล่วงหน้า และกระทั่งเรื่องใหญ่ระดับชาติอย่างข่าวการเกิดภัยพิบัติ จำพวกสึนามิหรือแผ่นดินไหว จะมีผู้แจ้งข่าวจำนวนมากกระจายอยู่รอบๆ บริเวณที่เกิดภัยพิบัติ แต่ละคนได้รับผลกระทบไม่เหมือนกันขึ้นอยู่กับพื้นที่ การวิเคราะห์ในเชิงภูมิศาสตร์สามารถนำมาใช้ประโยชน์ในการแจ้งเตือนภัยและให้ความช่วยเหลือผู้ประสบภัยได้

หากมองในแง่ของการวิเคราะห์ลักษณะการแพร่กระจายข่าวสารหรือพฤติกรรมของเครือข่ายสังคม จะเห็นได้ว่าข่าวสารแต่ละอย่างมีลักษณะการแพร่กระจายไม่เหมือนกัน ตัวอย่างเช่น ข่าวที่มาจากทีมงานของบริษัทหรือดาราน่าจะมีลักษณะกระจายออกไปจากจุดศูนย์กลางก่อน และเมื่อสำนักข่าวต่างๆ หรือผู้ที่มีผู้ติดตามมากๆ นำไปเผยแพร่ต่อก็จะกระจายออกไปจากแหล่งข้อมูลเหล่านั้นด้วย คำถามที่น่าสนใจคือ ข่าวสารใดหรือแหล่งข่าวใด มีการแพร่กระจายของข่าวสารลักษณะเป็นอย่างไรในเชิงภูมิศาสตร์ เช่น ความหนาแน่นของผู้สนใจต่อข่าวสารนั้นในพื้นที่หนึ่งๆ ระยะทางที่ข่าวสารไปถึงหรือความเร็วในการกระจายข่าว เป็นต้น

จากการพิจารณาผู้วิจัยมีความเห็นว่าข้อมูลในทวิตเตอร์เป็นสื่อที่เหมาะสมแก่การนำมาใช้วิเคราะห์หาความรู้ในเชิงภูมิศาสตร์มากที่สุดแหล่งหนึ่ง เพราะมีข้อมูลจำนวนมากเพียงพอต่อการนำไปศึกษาทั้งยังรองรับข้อมูลทางภูมิศาสตร์อย่างพิกัดตำแหน่งได้ และโครงงานของนายเจตพลก็ได้พิสูจน์แล้วว่าสามารถนำมาใช้งานจริงบนลูกโลกจำลองอย่างกูเกิลเอิร์ธได้อย่างเหมาะสม หากเพิ่มเติมความสามารถในการค้นหาและวิเคราะห์แบบออนไลน์ เพื่อให้สามารถค้นหาและตอบกลับผู้ใช้งาน ณ

ขณะนั้นได้ทันที จะช่วยให้แอพพลิเคชั่นนี้มีประสิทธิภาพและอาจทำให้เล็งเห็นถึงประโยชน์หรือความเป็นไปได้อื่นๆ สืบเนื่องไปอีกในอนาคต

ในที่นี้ผู้วิจัยจะเลือกสังเกตพฤติกรรมเฉพาะบางหัวข้อที่กำลังเป็นที่นิยมหรือน่าสนใจ ซึ่งมีหลักฐานจากแหล่งข่าวอื่นที่เชื่อถือได้มาทำการวิเคราะห์ เพื่อทำการเปรียบเทียบผลลัพธ์ว่ามีลักษณะและแนวโน้มไปในทางเดียวกันหรือไม่ เช่น ภูมิอากาศ, สภาพการจราจร หรือ ข่าวการก่อการร้าย เป็นต้น โดยพฤติกรรมดังกล่าวจะหมายถึง ลักษณะการเกาะกลุ่มหรือความหนาแน่นของข้อวิพากษ์วิจารณ์แต่ละภูมิภาค ณ พื้นที่ที่กำลังสนใจ, แนวโน้มของจำนวนผู้ใช้ทวิตเตอร์ที่สนใจต่อข่าวสารนั้นว่ามีแนวโน้มเพิ่มขึ้นหรือลดลง, การกระจายข่าวสารมีลักษณะเป็นลูกโซ่และขยายตัวออกไป หรือมีลักษณะเป็นข้อความโดด

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อพัฒนาเครื่องมือสำหรับวิเคราะห์ข้อมูลจากทวิตเตอร์ในเชิงพื้นที่และเวลาซึ่งจะนำไปศึกษาคุณลักษณะของข้อมูลที่ถูกเผยแพร่ในทวิตเตอร์รวมถึงพฤติกรรมของผู้ใช้ทวิตเตอร์ในประเทศไทย ได้แก่

1.2.1.1 การเรียงตัวของข้อความทวีตบนแผนที่สำหรับหัวข้อสนทนาที่แตกต่างกัน

1.2.1.2 การตอบสนองต่อเหตุการณ์หนึ่งๆ ที่เกิดขึ้นอย่างกะทันหัน

1.2.1.3 ความสนใจของผู้ใช้ทวิตเตอร์ต่อข่าวสารประเภทต่างๆ

1.2.1.4 ถิ่นที่อยู่อาศัยของผู้ใช้ทวิตเตอร์

1.3 ขอบเขตของการวิจัย

1.3.1 ข้อความที่นำมาวิเคราะห์เป็นข้อความที่มี Geolocation เท่านั้น

1.3.2 เครื่องมือสำหรับดึงข้อมูลจากทวิตเตอร์มาเก็บในฐานข้อมูล และแสดงผลบน Google Earth ได้จาก [7] และนำมาพัฒนาต่อ

1.3.3 การคำนวณวิเคราะห์จะใช้สมการ Haversine ในการหาระยะทางระหว่างพิกัด

1.3.4 จะแสดงผลลัพธ์ของการวิเคราะห์ผ่านหน้าเว็บควบคู่กับการแสดงผลบนกูเกิลเอิร์ธ

1.3.5 พิจารณาข้อมูลโดยปราศจากการไตร่ตรองในเชิงความหมาย

1.3.6 เลือกพิจารณาหัวข้อที่สนใจผ่านระบบค้นหาในฐานข้อมูล แล้วนำมาวิเคราะห์เพื่อให้ได้ความรู้ดังตัวอย่างต่อไปนี้

1.3.6.1 ระยะทางระหว่างข้อความแรกและข้อความที่อยู่ไกลที่สุด

1.3.6.2 ระยะทางที่ไกลที่สุดระหว่างข้อความที่เกี่ยวข้อง

1.3.6.3 ความเร็วในการแพร่กระจายข่าวสาร

1.3.6.4 จำนวนผู้ใช้งานและจำนวนการทวีตทั้งหมดต่อหัวข้อที่สนใจ

1.3.6.5 กราฟแสดงปริมาณข้อความต่อหนึ่งหน่วยเวลา ซึ่งจะแสดงให้เห็นแนวโน้มของข่าวสารว่ามีความน่าสนใจมากน้อยเพียงใด และมีกระแสที่ดีขึ้นหรือน้อยลง

1.3.6.6 ความหนาแน่นของผู้ใช้งานที่สนใจหัวข้อนั้นๆ ต่อพื้นที่ในเชิงภูมิภาค

1.4 ประโยชน์ที่คาดว่าจะได้รับ

ได้แอพพลิเคชั่นที่ใช้สังเกตพฤติกรรมของข้อความทวีตเกี่ยวกับข่าวสารต่างๆ ที่สื่อสารผ่านทางทวิตเตอร์ได้ ซึ่งพฤติกรรมของข้อมูลเหล่านั้นอาจถูกนำไปประยุกต์ใช้ในงานด้านอื่นๆ ต่อไป เช่น การตลาด, การวิจัยข้อมูล, สถิติ เป็นต้น

1.5 วิธีดำเนินการวิจัย

1.5.1 ศึกษาการทำงานของแอพพลิเคชั่นเครือข่ายสังคมบนกูเกิลเอิร์ธ

1.5.2 ศึกษาคุณลักษณะของข้อมูลที่ได้รับจากทวิตเตอร์

1.5.3 วิเคราะห์งานวิจัยที่เกี่ยวข้องรวมทั้งหลักการและเทคนิคต่างๆ ที่จะนำมาใช้

1.5.4 ออกแบบแอพพลิเคชั่นเพื่อเพิ่มฟังก์ชันการทำงานในส่วนของการวิเคราะห์ข้อมูลบนพื้นฐานของแอพพลิเคชั่นเครือข่ายสังคมบนกูเกิลเอิร์ธ

1.5.5 ประยุกต์ใช้งานกับกรณีศึกษาเพื่อทดสอบการทำงาน

1.5.6 วิเคราะห์ผลที่ได้จากการทดสอบ

1.5.7 สรุปผลและเรียบเรียงวิทยานิพนธ์

1.6 ผลงานตีพิมพ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้ตีพิมพ์และนำเสนอในการประชุมวิชาการดังนี้

1.6.1 บทความชื่อ “การวิเคราะห์ข้อมูลและแสดงผลข้อมูลเชิงพื้นที่และเวลาบนกูเกิลเอิร์ธ” [8]

1.6.1.1 ชื่อผู้แต่ง อิสราภรณ์ วิทยวิรานนท์ และ วีระ เหมืองสิน

1.6.1.2 ตีพิมพ์และนำเสนอในงานประชุมวิชาการชื่อ The 10th National Conference on Computing and Information Technology

(NCCIT2014) ซึ่งจัดขึ้นในวันที่ 8-9 พฤษภาคม 2557 ณ จ.ภูเก็ต ประเทศไทย

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 การทำเหมืองข้อความและการค้นคืนสารสนเทศ

2.1.1 ทวิตเตอร์ (Twitter)

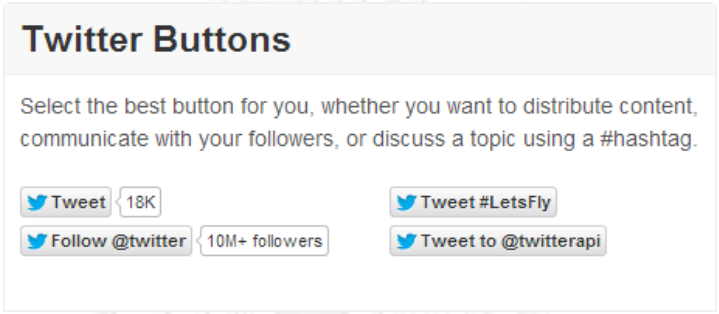
ทวิตเตอร์ [9] เป็นบริการไมโครบล็อกกิง (micro blogging) ซึ่งทำให้ผู้ใช้สามารถติดต่อเพื่อน ครอบครัว หรือที่ทำงานได้ตลอดเวลาผ่านคอมพิวเตอร์และสมาร์ตโฟน (smart phone) ผู้ใช้ทวิตเตอร์สามารถเขียนข้อความเพื่อบ่งบอกสถานะที่เรียกกันว่า ทวีต (Tweet) เพื่อประกาศเรื่องต่างๆ ให้เพื่อนหรือผู้ติดตามของผู้ใช้ได้รับรู้และสามารถทำการทวีตได้หลายๆ ครั้งในหนึ่งวัน ผู้ใช้อาจส่งข้อความเพื่อพูดคุยกับเพื่อนโดยใช้ “@user” Syntax ในการระบุและส่งข้อมูลให้ผู้รับ สำหรับการพูดคุยในลักษณะเป็นกลุ่มสามารถใช้ hashtags (#’s) เพื่อแท็ก (tag) ทวีตที่อยากติดตามได้ ซึ่งแน่นอนว่าไม่ว่าจะเป็นการระบุปลายทางหรือจะใช้แท็กผู้ใช้คนอื่นๆ ที่ติดตามอยู่ก็จะเห็นข้อความเหล่านั้นด้วยเสมอ ทำให้ข่าวสารแพร่กระจายไปอย่างรวดเร็วโดยอิสระ

นอกจากนี้ผู้ใช้ยังสามารถส่งต่อข้อความที่คนอื่นเขียนไปบนเว็บเพจของตัวเองได้ ซึ่งเรียกว่าการรีทวีต (Retweet) สำหรับการรีทวีตอาจแบ่งได้เป็น 2 แบบ คือ แบบส่งต่อโดยไม่มีการแก้ไขใดๆ แบบนี้เมื่อมีการรีทวีต ข้อความดั้งเดิมจะถูกเพิ่ม notation เข้าไปเป็น ‘RT @user msg’ อย่างอัตโนมัติ หากแต่จะไม่ปรากฏให้เห็นในหน้าเพจของผู้ใช้ กับอีกแบบหนึ่งคือทำการแก้ไขหรือเพิ่มเติมข้อความต้นฉบับ (Retweet Quote) ซึ่งแบบนี้ผู้ใช้จะต้องคำนึงถึงข้อจำกัดเรื่องความยาวตัวอักษรด้วยว่าข้อความรวมทั้งหมดนั้นจะต้องไม่เกิน 140 ตัวอักษรตามมาตรฐานของทวิตเตอร์ ดังนั้นในบางครั้งอาจจำเป็นต้องทำการลบข้อมูลบางส่วนออก

จะเห็นว่าโครงสร้างของทวิตเตอร์นั้นเอื้อต่อการกระจายข้อมูลออกไปตามเครือข่ายผู้ใช้ที่ปฏิสัมพันธ์กัน ทำให้คนจำนวนมากสามารถรับรู้ พูดคุย ถกเถียงเรื่องหนึ่งๆ ในเวลาเดียวกันได้ ถือว่าทวิตเตอร์เป็นเครือข่ายการสื่อสารที่รวดเร็วและกว้างขวางมากที่สุดแหล่งหนึ่งทีเดียว ด้วยปริมาณทวีตที่ทวิตเตอร์มีอยู่จึงเป็นสิ่งที่น่าสนใจสำหรับนักพัฒนาว่าข้อมูลบนระบบนั้นสามารถบอกอะไรที่เป็นความรู้แก่สาธารณะได้อีกบ้าง และเพื่อการนั้นทวิตเตอร์จึงได้นำเสนอการเข้าถึงคลังข้อมูลโดยแบ่งเป็นส่วนต่อประสานงานโปรแกรมประยุกต์ (Application Programming Interface: API) ออกเป็น 4 ส่วนใหญ่ๆ [10] โดยแต่ละส่วนจะนำเสนอฟังก์ชันการทำงานของทวิตเตอร์ ได้แก่

2.1.1.1 Twitter for Websites (TfW)

Twitter for Websites เป็นเครื่องมือที่ช่วยอำนวยความสะดวกให้กับนักพัฒนาเว็บไซต์ในการผสมผสานเว็บไซต์เข้ากับฟังก์ชันพื้นฐานของทวิตเตอร์ เช่น ปุ่มทวีต (tweet buttons) ซึ่งจะช่วยให้ผู้ใช้สามารถทวีตข้อความออกไป ปุ่มติดตาม (follow @twitter) เพื่อให้ผู้เยี่ยมชมเว็บไซต์สามารถติดตามข่าวสารของเว็บผ่านทางทวิตเตอร์ได้ เป็นต้น นอกจากนี้ยังมีส่วนของทวิตเตอร์แบบฝังตัว (Embedded Tweets) หมายถึงความสามารถในการฝังทวิตเตอร์ขนาดย่อมไว้ในเว็บไซต์



ภาพที่ 2.1 ตัวอย่างของ Twitter Buttons

ภาพที่ 2.2 ตัวอย่างของ Embedded Tweets

2.1.1.2 Search API

Search API จะอนุญาตให้นักพัฒนาสามารถค้นหาข้อมูลต่างๆ ในทวิตเตอร์ได้ โดยอาจค้นหาด้วยคีย์เวิร์ดหนึ่งๆ หรือด้วยชื่อผู้ใช้งาน และ API นี้จะอนุญาตให้อุปกรณ์ของผู้พัฒนาสามารถเข้าถึงข้อมูลเกี่ยวกับ Trends บนทวิตเตอร์ได้อีกด้วย

2.1.1.3 REST API

ทวิตเตอร์ได้เปิดโอกาสให้นักพัฒนาสามารถเข้าถึงข้อมูลพื้นฐานหลักๆ ของทวิตเตอร์ได้ เช่น Timeline, สถานะส่วนตัว และข้อมูลทั่วไปของผู้ใช้งาน ซึ่ง API นี้จะเหมาะกับงานที่ต้องการข้อมูลของผู้ใช้งาน อย่าง ชื่อ, วิธีการเข้าใช้, รูปภาพแทนตัว รวมถึงแผนภาพแสดงการติดตามของผู้ใช้งาน เป็นต้น ทั้งหมดนี้สามารถเรียกใช้ได้ด้วย REST API ไม่กี่ตัว นอกจากนี้ยังอนุญาตให้นักพัฒนาเข้าถึงหรือมีปฏิสัมพันธ์กับทวิตเตอร์ได้โดยตรง ทั้งการโพสข้อความการตอบกลับแม้แต่การรีทวีตหรือการทำงานอื่นๆได้อีกด้วย

2.1.1.4 Streaming API

ทวิตเตอร์ได้นำเสนอชุด API จำนวนหนึ่งเพื่อให้นักพัฒนาสามารถเข้าถึงกระแสข้อมูลทั่วไปของทวิตเตอร์แบบ real-time ได้ ซึ่งการติดตั้ง Streaming API ที่เหมาะสมจะทำให้ได้ข้อมูลของทวีตหรือข้อความที่เกิดขึ้นบนทวิตเตอร์ หรือเหตุการณ์อื่นๆ ทว่าปริมาณทวีตที่สามารถรับได้นั้นจะอยู่ที่ประมาณ 1% ถึง 40% ของทวีตที่เกิดขึ้นใหม่เท่านั้น และจะได้รับทวีตมากขึ้นหากระบุเงื่อนไขที่เจาะจงมากขึ้น [11]

สำหรับชุด API นี้ทวิตเตอร์ได้เสนอให้แก่นักพัฒนาทั้งหมด 3 แบบด้วยกัน คือ Public Stream, User Stream และ Site Stream ซึ่งในที่นี้จะเลือกใช้ Public Stream ซึ่งเป็น API ที่แบ่งปันข้อความทวีตบางส่วนกลับมาให้เพื่อเป็นตัวอย่างข้อมูลแก่นักพัฒนา ส่วน User Stream และ Site Stream เป็น API ที่ใช้ติดตามทวีตของผู้ใช้ทวิตเตอร์หนึ่งๆ โดย User Stream จะอนุญาตให้ติดตามผู้ใช้ทวิตเตอร์ได้เพียงหนึ่งคนเท่านั้น หากต้องการเรียกคืนข้อความทวีตจากหลายผู้ใช้ทวิตเตอร์ต้องเรียก Site Stream แทน

Public Stream [12] เป็น API ที่จะส่งตัวอย่างบางส่วนของ Twitter Firehose ซึ่งเป็นฐานข้อมูลขนาดใหญ่ของทวิตเตอร์ที่จัดเก็บสถานะทั้งหมดของผู้ใช้งาน ณ ปัจจุบันมาให้แก่ผู้พัฒนา ทำให้ API นี้เหมาะสมกับผู้พัฒนาโปรแกรมที่ต้องการข้อมูลจำนวนมาก เช่น โปรแกรมเกี่ยวกับเหมืองข้อมูล หรือ จะนำมาใช้ในการวิเคราะห์ข้อมูลก็ได้ โดย API นี้จะอนุญาตให้ผู้พัฒนาสามารถค้นหาข้อความด้วยคำค้นหาหรือคีย์เวิร์ดจำนวนมากได้ในเวลาเดียวกัน, ค้นหาข้อความที่มีคุณลักษณะทางภูมิศาสตร์ในเขตหนึ่งๆ หรือ จะร้องขอทวีตของผู้ใช้หนึ่งๆ จากทวิตเตอร์ก็ได้เช่นกัน

ทั้งนี้การร้องขอข้อมูลจากทวิตเตอร์ผ่าน Streaming API นั้นจะต้องการการเชื่อมต่อระยะยาวเพื่อให้สามารถรับข้อมูลจากทวิตเตอร์ได้อย่างต่อเนื่อง ซึ่งการสร้างการเชื่อมต่อนั้นจะต้องส่งคำ

ร้องขอเพื่อยืนยันตัวตน (Authentication, OAuth) กับทวิตเตอร์เสียก่อน เมื่อได้เชื่อมต่อผ่าน HTTP โปรโตคอลแล้วจึงส่งคำร้องขอข้อมูลไปอีกครั้งหนึ่งด้วย HTTP method GET หรือ POST ตามความต้องการในการใช้งาน โดย endpoint มีให้เลือกใช้ 3 แบบ คือ POST statuses/filter, GET statuses/sample และ GET statuses/firehose และในที่ได้เลือกใช้แบบ POST statuses/filter

Endpoint statuses/filter นั้นจะส่งคืนทวีตที่มีสถานะสาธารณะ (Public statuses) แก่ต้นทางโดยคัดกรองจากพารามิเตอร์ที่ส่งไปร้องขอ โดยพารามิเตอร์หรือตัวบ่งชี้ข้อมูลเป้าหมายซึ่งทวิตเตอร์จัดสรรไว้ดังตารางที่ 2.1

ตารางที่ 2.1 ชุดพารามิเตอร์ของ Endpoint Filter

พารามิเตอร์	คำอธิบาย	ประเภท
Follow	การใช้ User ID ในการติดตามข้อความทวีต อนุญาตให้ติดตามได้หลาย User ID โดยอาศัย comma (,) คั่นระหว่าง ID	Condition
Track	การใช้คีย์เวิร์ดในการสืบค้น อนุญาตให้ค้นหาได้หลายๆ คีย์เวิร์ด โดยอาศัย comma (,) คั่นระหว่างคีย์เวิร์ดที่ใช้ในการค้นหา	Condition
Location	ระบุขอบเขต Bounding box (BBOX) ในการค้นหาทวีต	Condition
Delimited	ค่าที่เป็นไปได้คือ length ใช้เพื่อขอให้ทวิตเตอร์ระบุความยาวของ String ของข้อความทวีตที่ส่งกลับคืนว่ามีจำนวนกี่ไบต์ เพื่อบ่งบอกว่าต้นทางจะได้รับข้อความขนาดเท่าไหร่	Optional
Stall_warning	ค่าที่เป็นไปได้คือ true เพื่อขอรับคำแจ้งเตือนจากทวิตเตอร์เมื่อต้นทางเริ่มมีปัญหาในการติดต่อกับทวิตเตอร์ ซึ่งทวีตจะถูกกักเก็บไว้เพื่อรอส่งใหม่ให้ต้นทาง หลังจากต้นทางทำการติดต่อเข้าไปใหม่ แต่หากว่าทวีตถูกกักเก็บไว้จนเต็มพื้นที่ที่ทวิตเตอร์จัดเตรียมไว้ให้ ทวิตเตอร์จะทำการตัดการเชื่อมต่อด้วยตนเอง	Optional
count	กรณีที่การเชื่อมต่อสิ้นสุดลงจะมีการเก็บ backfill message ลงในคิวของทวิตเตอร์ หากต้นทางหรือผู้ใช้ต้องการร้องขอข้อความส่วนที่หายไปสามารถส่งพารามิเตอร์นี้ได้ โดยจะมีค่าในช่วง 1 ถึง 150000 หรือ -1 ถึง -150000 โดยค่าบวกจะหมายถึงให้ทำการรอรับ live message ต่อเมื่อรับ backfill message ครบแล้ว ส่วนค่าลบทวิตเตอร์จะทำการตัดการเชื่อมต่อเมื่อส่ง backfill message ให้ผู้ใช้ครบถ้วน	Optional

นอกจากคุณลักษณะส่วนตัวที่ทำให้การสื่อสารกระชับรวดเร็วของทวิตเตอร์ และชุดคำสั่งที่เอื้อประโยชน์แก่นักพัฒนา อีกเหตุผลหนึ่งที่ทำให้ทวิตเตอร์เป็นแหล่งข้อมูลที่น่าสนใจก็คือความสามารถในการแนบ Geolocation [13] ไปบนทวีตแต่ละทวีต ซึ่งเป็นการระบุพิกัดของโลกบน

แผนที่ผ่านอุปกรณ์อิเล็กทรอนิกส์ที่สามารถเชื่อมต่อกับอินเทอร์เน็ตได้ บางครั้ง Geolocation ก็ถูกกล่าวถึงและเชื่อมโยงไปยัง GPS (Global Positioning System) ซึ่งเป็นระบบบอกพิกัดตำแหน่งบนโลกที่อาศัยการคำนวณพิกัดผ่านดาวเทียมอยู่บ่อยครั้ง แต่แท้จริงแล้วการระบุตำแหน่งอาจนำมาซึ่งวิธีอื่นๆนอกเหนือจาก GPS ก็ได้เช่นกัน เช่น IP address เป็นต้น ดังนั้นการเลือกใช้ชุด Public stream API ในงานวิจัยนี้จะใช้ endpoint filter โดยคัดกรองด้วย Location

การคัดกรองด้วย Location [14] จะค้นหาทวีตที่อยู่ในกรอบของพื้นที่ซึ่งระบุเอาไว้ใน BBOX อันประกอบด้วยพิกัดมุมซ้ายล่าง (Bottom left: BL) และมุมขวาบน (Top right: TR) เรียงตามลำดับ [lonBL, latBL, lonTR, latTR] filter นี้จะคัดกรองเฉพาะทวีตที่เพิ่งเกิดขึ้นใหม่ โดยการค้นหาจะเริ่มจาก coordinates field ของข้อความว่ามีการระบุพิกัดในข้อความหรือไม่ หากมีและตกอยู่ในพื้นที่ของ BBOX ทวิตเตอร์ก็จะส่งคืนให้ผู้ร้องขอ แต่หากไม่มีการแนบพิกัดไว้ใน field ดังกล่าว ทวิตเตอร์จะตรวจสอบ place field แทน หากภูมิภาคที่ระบุไว้ตรงกับพื้นที่ของ BBOX ก็จะส่งคืนเช่นเดียวกัน ในทางกลับกันหากไม่เจอทั้งพิกัดใน coordinate และ place field ทวิตเตอร์จะข้ามทวีตนั้นไป และเนื่องด้วยข้อความรีทวีตยังไม่สนับสนุนการแนบพิกัดลงในข้อความดังนั้นข้อความรีทวีตนี้จะไม่ตกเงื่อนไขใดๆ ของ filter เสมอ แม้ว่าทวีตต้นจะมีการระบุพิกัดก็ตาม เช่น locations=-122.75,36.8,-121.75,37.8 จะกรองเอาทวีตที่อยู่ในพื้นที่ของ San Francisco

อย่างไรก็ดีทวิตเตอร์ที่มีความสามารถในการรองรับข้อมูลเชิงภูมิศาสตร์นี้จะแนบพิกัดหรือที่อยู่ ณ ขณะที่ผู้ใช้ทวิตเตอร์ทำการทวีตไปกับข้อความได้ ซึ่งในแง่ของความเป็นส่วนตัวทวิตเตอร์ได้กำหนดค่าเริ่มต้นเป็นโหมดออฟเอาไว้ หากผู้ใช้งานต้องการเปิดใช้งานก็สามารถเข้าไปกำหนดค่าต่างๆ ในส่วนของการตั้งค่า (Setting) ได้ด้วยตนเอง [15] และในบางกรณีที่ไม่อยากให้ส่ง Geolocation เข้าไปในบางทวีตผู้ใช้ยังสามารถลบพิกัดของตนได้ก่อนส่งขึ้นไปบนหน้าเพจ หรือกำหนดว่าที่อยู่ใดเป็นที่อยู่ส่วนตัวไม่ขอให้เปิดเผยบนทวิตเตอร์ก็ทำได้เช่นเดียวกัน

2.1.2 Tweepy

Tweepy [16] เป็นไลบรารี่ภาษาสคริปต์ไพธอน (Python) เพื่อใช้ติดต่อกับทวิตเตอร์และเรียกใช้ API ของทวิตเตอร์ ซึ่งรวมถึง Streaming API ที่จะใช้ในงานวิจัยนี้ด้วย โดยการเรียกใช้งานจะต้องติดตั้ง package Tweepy ก่อน แล้วจึงทดสอบโดยการ import Tweepy นี้เข้ามาใน code ตัวอย่างการใช้งานในที่นี้คือการอัพเดตสถานะของผู้ใช้หนึ่งๆ ตามรูปที่ 2.3 โดยจะเริ่มต้นเชื่อมต่อไป

ยังทวิตเตอร์ด้วย OAuth ก่อนเพื่อยืนยันตัวตนแล้วจึงเรียกใช้ฟังก์ชัน update_status ของ tweepy เพื่ออัพเดตสถานะ

2.1.3 ฐานข้อมูล MySQL

เมื่อกระแสข้อมูลไหลเข้ามาในระบบ การจัดเก็บข้อมูลเพื่อรอวิเคราะห์ต่อไปจึงเป็นเรื่องสำคัญ ในงานวิจัย [7] ได้เลือกใช้ SQLite database ซึ่งเป็นฐานข้อมูลขนาดเล็กมาใช้งานเพื่อทำการทดลองระยะสั้น จึงอาจไม่เหมาะกับการนำมาใช้เก็บข้อมูลขนาดใหญ่สำหรับงานวิจัยนี้ สำหรับ MySQL database [17] ที่เป็นฐานข้อมูลแบบ open source ทั้งยังมีศักยภาพสูงจึงเป็นตัวเลือกที่ดีกว่า เพราะสามารถรองรับจำนวนข้อมูลขนาดใหญ่และการเข้าใช้งานจำนวนมาก (concurrents) ได้ดี ทั้งยังง่ายต่อการพัฒนาหรือปรับแต่งประสิทธิภาพในอนาคตอีกด้วย

MySQL 5.1 ได้เพิ่มความสามารถในส่วนของการจัดการข้อมูลแบบเป็น partition เข้ามาด้วย โดยความสามารถนี้จะช่วยเอื้อการทำงานในส่วนของการจัดเก็บถาวรของข้อมูลที่สนใจได้ดีมากยิ่งขึ้น ดังนั้นในงานวิจัยนี้จึงเลือกติดตั้ง MySQL 5.1 เพื่อใช้กับแอพพลิเคชั่น โดยข้อมูลของส่วนพิกัดภูมิศาสตร์จะจัดเก็บในรูปของ String

```
#!/usr/bin/python -x

#============================Import module part
import tweepy
import time,os

#============================Define parameters
# Get these values from your application settings
CONSUMER_KEY = '[redacted]'
CONSUMER_SECRET = '[redacted]'

# Get these values from the "My Access Token" link located in the
# margin of your application details, or perform the full OAuth
# dance
ACCESS_TOKEN = '[redacted]'
ACCESS_TOKEN_SECRET = '[redacted]'


#============================Define functions
def OAuth():

    auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
    auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)


    # Note: Had you wanted to perform the full OAuth dance instead of using
    # an access key and access secret, you could have uses the following
    # four lines of code instead of the previous line that manually set the
    # access token via auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECR
    #
    # auth_url = auth.get_authorization_url(signin_with_twitter=True)
    # webbrowser.open(auth_url)
    # verifier = raw_input('PIN: ').strip()
    # auth.get_access_token(verifier)

    return auth


def main():
    # Call method to establish the connection to twitter

    auth = OAuth()
    api = tweepy.API(auth)
    print "test"
    api.verify_credentials()
    dateTime = time.strftime("%Y/%m/%d %H:%M:%S")
    text = "Tweet this msg from my app at %s" %dateTime
    api.update_status(text)

    while True:
        try:
                print "The connection is established"
        except Exception, e:
                print "Error"
    else:
        print "Error to connect"

#============================Real process

main()
```

ภาพที่ 2.3 ตัวอย่างการใช้งาน Tweepy

2.2 ข้อมูลสารสนเทศเชิงพื้นที่และเวลา

2.2.1 สมการ Haversine

การหาระยะทางระหว่างพิกัดสองตำแหน่งบนโลกจำเป็นจะต้องใช้หลักการทางภูมิศาสตร์เข้ามาช่วยเนื่องจากโลกไม่ได้มีลักษณะเป็นทรงแบนหรืออยู่ในแนวระนาบตลอด ดังนั้นหากจะเลือกใช้การหาระยะทางแบบยุคลิดในที่นี้จึงอาจไม่เหมาะสมนัก การประมาณระยะทางควรจะคำนึงถึงแนวโค้งของผิวโลกด้วยเพราะโลกมีลักษณะเป็นทรงกลม และแม้ว่าความจริงแล้วผิวโลกอาจไม่ได้ราบเรียบสม่ำเสมอตลอด การตั้งสมมติฐานว่าทุกจุดบนโลกมีรัศมีเท่ากันก็ยังคงสามารถทำได้ เพราะความแตกต่างที่เกิดขึ้นระหว่างผิวโลกกับส่วนที่ผิดปกติ (อาจจะต่ำกว่าหรือสูงกว่าพื้นผิวทั่วไป) นั้นมีขนาดเล็กมากเมื่อเทียบกับรัศมีของโลก

ด้วยเหตุผลข้างต้นในที่นี้จะใช้วิธี Great-Circle distance เพื่อหาระยะทางที่สั้นที่สุดระหว่างจุดสองจุดบนผิวโลก และแน่นอนว่าการหาระยะทางที่สั้นที่สุดซึ่งอ้างอิงแนวโค้งของผิวโลกนั้นมีอยู่หลายสูตรหลายวิธี ทั้งนี้จะขอนำเสนอการหาระยะทางด้วยสมการ Haversine ซึ่งเป็นสมการคณิตศาสตร์ที่สำคัญในระบบนำร่อง และคำนวณระยะทางจากพิกัดละติจูดลองจิจูดที่สนใจ [18]

Haversine Formula:

$$h = haversin\left(\frac{d}{r}\right)$$

$$haversin\left(\frac{d}{r}\right) = haversin(\emptyset_2 - \emptyset_1) + cos(\emptyset_1)\,cos(\emptyset_2)\,haversin(\lambda_2 - \lambda_1)$$

โดย Haversin จะหมายถึง Haversine function

$$haversin(\theta) = sin^2\left(\frac{\theta}{2}\right) = \frac{1 - cos\,\theta}{2}$$

และ

r คือ รัศมีของโลก (รัศมีเฉลี่ย = 6,371 กิโลเมตร)

d คือ ระยะทางระหว่างจุดสองจุด

$\emptyset_1, \emptyset_2$ คือ ละติจูดของจุดที่ 1 และ ละติจูดของจุดที่ 2 ตามลำดับ

λ_1, λ_2 คือ ลองจิจูดของจุดที่ 1 และ ลองจิจูดของจุดที่ 2 ตามลำดับ

ดังนั้นจากสมการ Haversine ข้างต้นจะทำให้เราสามารถหาระยะทางได้โดยการทำ Inverse Haversine function ดังนี้

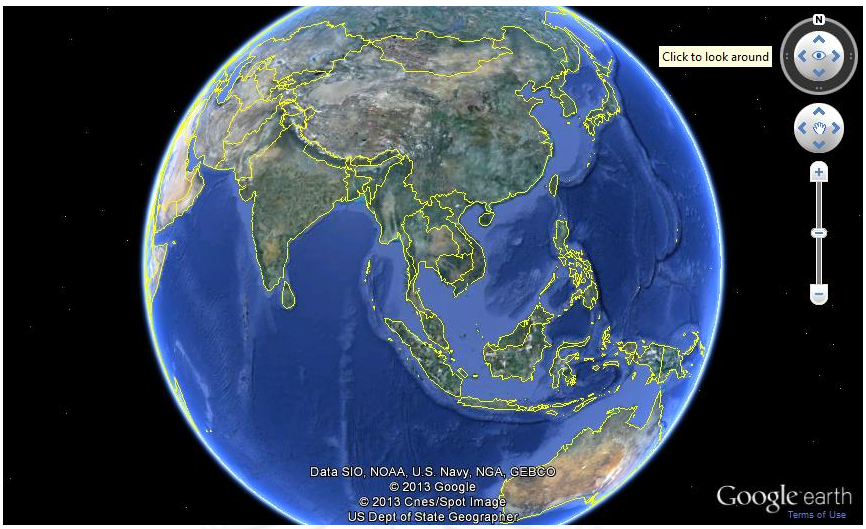
$$d = rhaversin^{-1}(h) = 2r\ arcsin\left(\sqrt{h}\right)$$

2.2.2 กูเกิลเอิร์ธ (Google Earth)

กูเกิลเอิร์ธเป็นซอฟต์แวร์ที่ถูกพัฒนาโดยบริษัทกูเกิล เพื่อนำเสนอเส้นทางและภูมิศาสตร์ต่างๆ บนโลกรวมทั้งระบบ GIS (Geographic Information System) ในลักษณะของภาพ 3 มิติ ทั้งสามารถค้นหาเมือง หรือสถานที่ใดๆ ในโลกผ่านคอมพิวเตอร์ส่วนตัว (personal computer) หรือกระทั่งโทรศัพท์มือถือสมาร์ตโฟนได้ แสดงภาพส่วนติดต่อกับผู้ใช้งานดังรูปที่ 2.4

กูเกิลเอิร์ธ [19] จะใช้ข้อมูลจากภาพถ่ายทางอากาศของ U.S. public domain และภาพถ่ายของ Keyhole มาดัดแปลงร่วมกับระบบแผนที่ของกูเกิลจาก Google Mapss รวมทั้งการทำงานร่วมกับ Google Local เพื่อค้นหารายชื่อสถานที่ เช่น ร้านขายของ ธนาคาร ร้านอาหาร และปั๊มน้ำมันในแผนที่ได้ โดยนำแผนที่มาซ้อนทับลงบนตำแหน่งที่ต้องการซึ่งสามารถหาได้จากบ้านเลขที่, ลองจิจูด-ละติจูด ซึ่งทำงานผ่านรูปแบบภาษาของ KML (Keyhole Markup Language) ที่จะกล่าวต่อไปในหัวข้อ 2.2.4

ทั้งนี้จะต้องมีกูเกิลเอิร์ธปลั๊กอิน (Google Earth Plug-in) และ JavaScript API ช่วยผนวกกูเกิลเอิร์ธเข้ากับระบบหรือเว็บแอพพลิเคชั่นที่ต้องการใช้งาน ซึ่งได้จัดสรร API ในการวาดเส้น ปักหมุด การทำโมเดลต่างๆ หรือการโหลด KML ไฟล์เอาไว้ด้วย



ภาพที่ 2.4 ตัวอย่างส่วนติดต่อกูเกิลเอิร์ธสำหรับผู้ใช้งาน

2.2.3 Google Earth API

Google Earth API [20] เป็นชุดคำสั่งที่ช่วยในการควบคุมกูเกิลเอิร์ธ โดยการเรียกใช้งานนั้นจะเริ่มจากการโหลด JavaScript API เข้ามาก่อน คือ <script type="text/javascript" src="https://www.google.com/jsapi"></script> เพื่อให้สามารถเรียกใช้ API ต่างๆได้ และ API เบื้องต้นที่นำมาใช้งานได้แก่

- google.load : ทำการโหลด Earth มอดูลมาเก็บไว้ใน google.earth namespace
- google.earth.createInstance : ระบุ DIV element ที่ต้องการสร้าง instance นี้ลงไป และระบุฟังก์ชันที่จะทำงานต่อเมื่อได้รับ success หรือ failed กลับมา
- getNavigationControl : ติดตั้งเครื่องมือในการขยายเข้า/ออกบนแผนที่
- getLayerRoot : ระบุว่าต้องการให้แสดงถนนหรือเส้นขอบประเทศบนแผนที่หรือไม่
- getWindow : จะถูกเรียกใช้เมื่อสามารถสร้าง instance ได้สำเร็จ หน้าแผนที่กูเกิลเอิร์ธจะปรากฏขึ้นมา
- createLookAt : ใช้ในการปรับเปลี่ยนมุมมองให้กล้องไปยังพื้นที่ที่ต้องการ โดยจะทำการระบุจุดศูนย์กลางด้วยละติจูด-ลองจิจูดและกำหนดความสูงของกล้องกับพื้นดินว่าห่างกันในระยะกี่เมตร
- getView().setAbstractView(lookAt) : ใช้ร่วมกับ createLookAt เพื่อเปลี่ยนมุมมองไปยัง view ที่ระบุจาก createLookAt

- getFeatures().appendChild(placemark) : โหลด KML หรือ KMZ ไฟล์เพื่อปักหมุด Placemark ลงบนแผนที่
- setBalloon : ใช้เปิด/ปิดบอลลูนข้อความบนแผนที่ของหมุด
- google.earth.addEventListener : ใช้จัดทำ listener เพื่อรอรับ event ใดๆ ที่เกิดขึ้นกับ object บนแผนที่และทำการเรียกฟังก์ชันที่เหมาะสม

แสดงตัวอย่างการใช้งาน Google Earth API เบื้องต้นตามโค้ดในรูปที่ 2.5 และผลลัพธ์ตามรูปที่ 2.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                        <title>Sample Google Earth</title>
                        <script type="text/javascript" src="https://www.google.com/jsapi"></script>
                        <script type="text/javascript">
                                var ge;
                                google.load("earth", "1");

                                function init() {
                                         google.earth.createInstance('map3d', initCB, failureCB);
                                }

                                function initCB(instance) {
                                    ge = instance;
                                ge.getWindow().setVisibility(true);
                                        //showLoading();
                                        //First load would be set at Thailand
                                        var lookAt = ge.getView().copyAsLookAt(ge.ALTITUDE_RELATIVE_TO_SEA_FLOOR);
                                        lookAt.setLatitude(12.93627753530871);
                                lookAt.setLongitude(101.9579417154536);
                                        lookAt.setAltitude(0);
                                        lookAt.setHeading(0.001725095909370177);
                                        lookAt.setTilt(0.03671988362490631);
                                        lookAt.setRange(2800000);               //To set height from sea floor
                                        lookAt.setAltitudeMode(ge.ALTITUDE_RELATIVE_TO_SEA_FLOOR);
                                ge.getView().setAbstractView(lookAt);
                                // add a navigation control
                                ge.getNavigationControl().setVisibility(ge.VISIBILITY_AUTO);
                                // add some layers
                                        ge.getLayerRoot().enableLayerById(ge.LAYER_BORDERS, true);
                                        ge.getLayerRoot().enableLayerById(ge.LAYER_ROADS, true);


                                }

                                function failureCB(errorCode) {
                                }

                                google.setOnLoadCallback(init);

                        </script>
        </head>
        <body>
                        <div id="map3d" style="height: 400px; width: 600px;"></div>
        </body>
</html>
```


ภาพที่ 2.5 ตัวอย่างการใช้งาน Google Earth API

ภาพที่ 2.6 แผนที่กูเกิลเอิร์ธผลลัพธ์จากตัวอย่างการใช้งาน Google Earth API

2.2.4 KML (Keyhole Markup Language)

KML เป็นไฟล์รูปแบบหนึ่งซึ่งมีลักษณะเป็น XML (eXtensible Markup Language) notation ถูกพัฒนาขึ้นในปี 2004 โดย Keyhole, Inc [21] เพื่อใช้กับแผนที่โลกอย่าง กูเกิลเอิร์ธ, กูเกิลแมพส์ (Google Maps) หรือ กูเกิลแมพส์สำหรับมือถือ สำหรับแสดงข้อมูลภูมิประเทศบนแผนที่ทั้งบนระบบ 2 มิติและ 3 มิติ และต่อมาได้กลายเป็นมาตรฐานสากลของ Open Geospatial Consortium ในปี 2008 ซึ่งการปรับแต่ง KML file นี้ทำให้สามารถแสดงจุด เส้น ภาพรูปหลายเหลี่ยม ข้อความหรือแม้แต่โครงสร้างโมเดลบนแผนที่ได้อย่างเหมาะสมตามความต้องการของผู้ใช้งาน

ตามจริงแล้วความสามารถบางอย่างสามารถสร้างได้ตรงๆ บนส่วนเชื่อมต่อกูเกิลเอิร์ธ (Google Earth User Interface) แต่บางอย่างจำเป็นจะต้องสร้างผ่าน KML file แต่ในที่นี้ความสามารถหลักที่จะถูกนำมาใช้คือ การสร้าง Placemark และการสร้างเส้นเชื่อมต่อระหว่าง Placemark

ในการสร้าง Placemark โครงสร้างต้องประกอบด้วยส่วนต่างๆดังนี้

- XML Header ซึ่งระบุเวอร์ชั่นของ XML จะเป็นบรรทัดแรกเสมอ
- ประกาศ KML namespace ในบรรทัดที่สอง

- Placemark object ซึ่งประกอบด้วย
    - Name ชื่อที่ใช้ระบุ Placemark
    - Description ซึ่งจะไปปรากฏอยู่ในบอลลูนของ Placemark
    - Point เพื่อระบุตำแหน่งของ Placemark บนแผนที่ โดยปกติจะใช้ลองจิจูด-ละติจูดในการบอกตำแหน่ง หรือจะระบุอัลติจูด (altitude) สำหรับความสูงด้วยก็ได้เช่นกัน

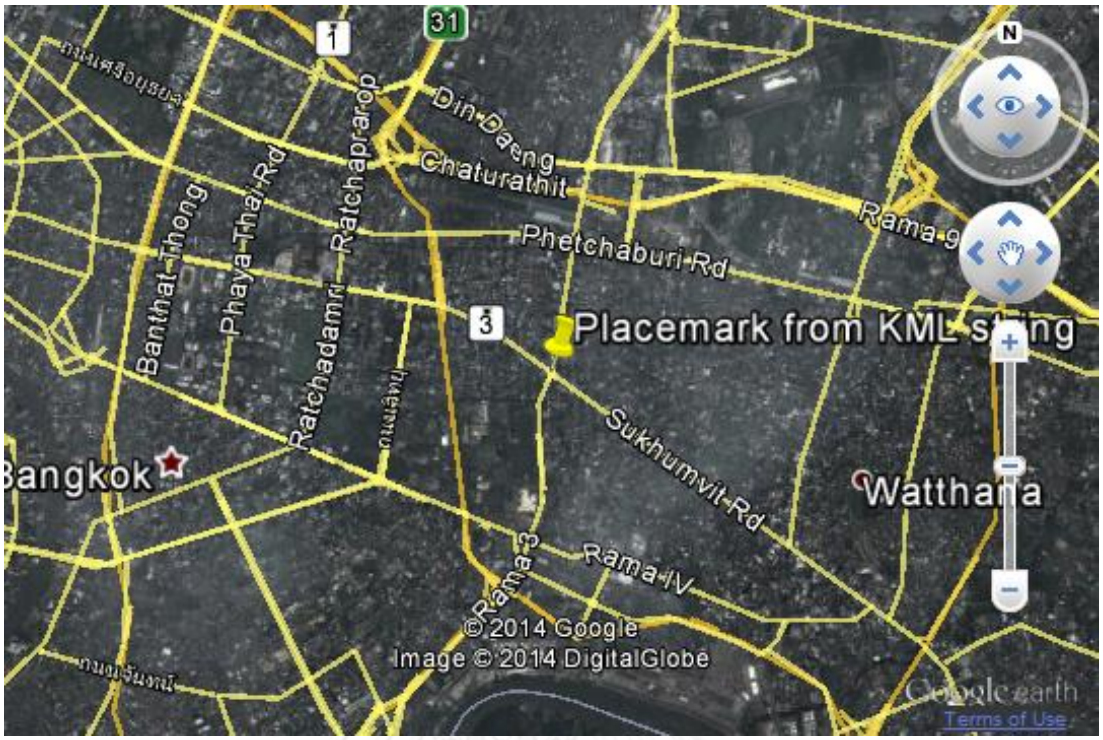
สำหรับการสร้าง Line ระหว่าง Placemark จะมีส่วนประกอบหลักเหมือนกันกับ Placemark ยกเว้นส่วนของ Placemark object ที่จะอาศัย LineString element ในการกำหนดเส้นเชื่อมขึ้นมา

- Placemark object ซึ่งประกอบด้วย
    - LineString สำหรับสร้างเส้นเชื่อมระหว่าง Placemark โดยมีส่วนประกอบสำคัญ คือ extrude (ระบุว่าต้องการให้เส้นลู่เข้าหาพื้นโลกหรือไม่), tessellate (ปรับให้เป็นเส้นประหรือไม่) และ coordinates (อนุญาตให้เชื่อมต่อระหว่างจุดใดๆ)

ทั้งนี้เมื่อรวบรวมองค์ประกอบต่างๆ ครบตามต้องการแล้วจึงบันทึกไฟล์ให้เป็น KML หรือ KMZ สำหรับกรณีต้องการบีบอัดไฟล์ ตัวอย่างการปักหมุดด้วย KML string เป็นดังรูปที่ 2.7 และผลลัพธ์ที่ได้ดังรูปที่ 2.8

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                        <title>Sample Google Earth</title>
                        <script type="text/javascript" src="https://www.google.com/jsapi"></script>
                        <script type="text/javascript">
                                var ge;
                                google.load("earth", "1");

                                function init() {
                                        google.earth.createInstance('map3d', initCB, failureCB);
                                }

                                function initCB(instance) {
                                    ge = instance;
                                    ge.getWindow().setVisibility(true);
                                    //showLoading();
                                    //First load would be set at Thailand

                                    var lookAt = ge.getView().copyAsLookAt(ge.ALTITUDE_RELATIVE_TO_SEA_FLOOR);
                                    lookAt.setLatitude(13.736655);
                                    lookAt.setLongitude(100.561199);
                                    lookAt.setAltitude(0);
                                    lookAt.setHeading(0.001725095909370177);
                                    lookAt.setTilt(0.03671988362490631);
                                    lookAt.setRange(10000);               //To set height from sea floor
                                    lookAt.setAltitudeMode(ge.ALTITUDE_RELATIVE_TO_SEA_FLOOR);

                                    ge.getView().setAbstractView(lookAt);
                                    // add a navigation control
                                    ge.getNavigationControl().setVisibility(ge.VISIBILITY_AUTO);
                                    // add some layers
                                    ge.getLayerRoot().enableLayerById(ge.LAYER_BORDERS, true);
                                    ge.getLayerRoot().enableLayerById(ge.LAYER_ROADS, true);
                                    // Create the placemark.
                                    var kmlString = ''
                                                + '<?xml version="1.0" encoding="UTF-8"?>'
                                                + '<kml xmlns="http://www.opengis.net/kml/2.2">'
                                                + '<Document>'
                                                + '  <Placemark>'
                                                + '    <name>Placemark from KML string</name>'
                                                + '    <Point>'
                                                + '      <coordinates>100.561199,13.736655,0</coordinates>'
                                                + '    </Point>'
                                                + '  </Placemark>'
                                                + '</Document>'
                                                + '</kml>';

                                    var kmlObject = ge.parseKml(kmlString);
                                    ge.getFeatures().appendChild(kmlObject);
                                }

                                function failureCB(errorCode) {
                                }

                                google.setOnLoadCallback(init);

                        </script>
        </head>
        <body>

                        <div id="map3d" style="height: 400px; width: 600px;"></div>
        </body>
</html>
```

ภาพที่ 2.7 ตัวอย่างการใช้ KML string ในการปักหมุดบนแผนที่



ภาพที่ 2.8 ปักหมุดบนแผนที่ผลลัพธ์จากตัวอย่างการใช้ KML

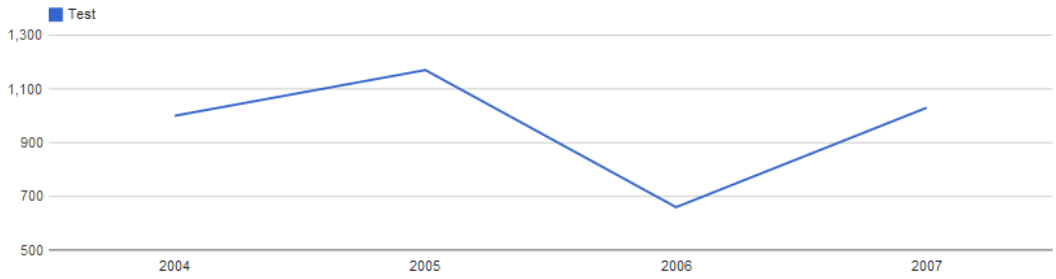
2.2.5 แผนภูมิกูเกิล (Google Chart)

แผนภูมิกูเกิล [22] ได้เปิดโอกาสให้นักพัฒนาสามารถแสดงข้อมูลที่มีอยู่ในหลากหลายรูปแบบตั้งแต่แผนภูมิเส้นพื้นฐานไปจนถึงแผนภาพต้นไม้แบบซับซ้อน ซึ่งทั้งหมดนั้นสามารถเรียกใช้งานได้ผ่าน JavaScript ที่จะถูกฝังอยู่ในเบราว์เซอร์ของฝั่งผู้ใช้งาน โดยอนุญาตให้ผู้พัฒนาโหลดไลบรารี่ของแผนภูมิกูเกิล แล้วรวบรวมข้อมูลเพื่อมาทำแผนภูมิ จากนั้นจึงเลือกรูปแบบที่ต้องการเพื่อสร้างแผนภูมิหนึ่งๆ ขึ้นมา แผนภูมินั้นจะถูกระบุด้วย id ที่ตั้งขึ้นเองโดยผู้พัฒนา แล้วจะถูกเรียกไปแสดงบนหน้าเว็บเพจเมื่อมีการเรียก id นั้นๆใน tag

สำหรับแผนภูมิเส้นหรือ Line chart นั้นถือเป็นการแสดงข้อมูลขั้นพื้นฐานที่เข้าใจง่ายที่สุด เพราะแสดงให้เห็นถึงแนวโน้มที่ชัดเจนของข้อมูล เมื่อมีการเปรียบเทียบระหว่างหัวข้อสองหัวข้อหรือมากกว่าแผนภูมิจะช่วยให้เห็นถึงความแตกต่างได้อย่างดี แผนภูมิเส้นจึงเป็นตัวเลือกที่เหมาะสมในการประยุกต์ใช้ในงานวิจัย ตัวอย่างการใช้งานเป็นไปตามรูปที่ 2.9 และ 2.10

```
<html>
  <head>
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
      google.load("visualization", "1", {packages:["corechart"]});
      google.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Year', 'Test'],
          ['2004',  1000],
          ['2005',  1170],
          ['2006',  660],
          ['2007',  1030]
        ]);
        var stepUp = 86400; // 1 day
        var options = {
                        fontSize:12,
                        'width':"100%",
                        'height':"100%" ,
                        'chartArea':{left:40,top:40, width:"100%"},
                        'legend': {'position': 'top'}
                      };

        var chart = new google.visualization.LineChart(document.getElementById('chart_div'));
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="chart_div" style="width: 900px; height: 300px;"></div>
  </body>
</html>
```

ภาพที่ 2.9 ตัวอย่างโค้ดในการสร้างแผนภูมิเส้นด้วยแผนภูมิกูเกิล



ภาพที่ 2.10 แผนภูมิเส้นผลลัพธ์จากตัวอย่างการสร้างแผนภูมิเส้นด้วยแผนภูมิกูเกิล

2.2.6 JavaScript

JavaScript [23] เป็นภาษาโปรแกรมประเภทหนึ่งที่เรียกว่าสคริปต์ถูกพัฒนาโดย Brendan Eich ซึ่งจะทำงานบนเบราว์เซอร์ฝั่งไคลเอนต์ เพื่อเพิ่มความสามารถในการตอบสนองแก่เว็บเพจที่เมื่อก่อน HTML (Hyper Text Markup Language) นำเสนอได้แต่ข้อมูลหรือเอกสารที่มีเนื้อหาคงที่ (static) และด้วยเบราว์เซอร์เวอร์ชั่นใหม่ๆต่างรองรับการแปลภาษาสคริปต์นี้จึงทำให้การเขียนเว็บเพจโดยใส่ JavaScript เพื่อเพิ่มลูกเล่นต่างๆ เป็นที่นิยมอย่างรวดเร็ว

ในด้านความสามารถการประมวลผลข้อมูลด้วยภาษานี้จะขึ้นอยู่กับคุณสมบัติของเครื่องคอมพิวเตอร์ฝั่งไคลเอนต์เป็นหลัก เพราะครั้งแรกที่เบราว์เซอร์ทำการร้องขอหน้าเว็บเพจมายังเซิร์ฟเวอร์ เซิร์ฟเวอร์จะทำการส่งคืนหน้าเว็บที่เหมาะสมพร้อมฝัง JavaScript กลับไปให้ ทางเบราว์เซอร์จะจดจำโปรแกรมสคริปต์เอาไว้และเมื่อมีการเรียกข้อมูลอื่นๆ ไปที่เซิร์ฟเวอร์ ผลลัพธ์ที่ได้กลับมาจะถูกประมวลผลหรือวิเคราะห์ตามแต่เงื่อนไขในโปรแกรมสคริปต์ก่อนแสดงผลให้ผู้ใช้ได้เห็น ด้วยเหตุนี้จึงสามารถแบ่งเบาภาระของเครื่องเซิร์ฟเวอร์บางส่วนไปให้ฝั่งไคลเอนต์จัดการต่อได้

2.2.7 jQuery library

jQuery เป็น Javascript library หนึ่งที่บรรจุฟังก์ชั่นการทำงานต่างๆ เอาไว้ช่วยอำนวยความสะดวกให้กับผู้พัฒนาไม่ต้องเขียนโค้ดยาวๆ สำหรับการทำงานเหล่านั้น ทั้งยังมีความสามารถในการทำงานแบบ Asynchronous JavaScript and XML (AJAX) ซึ่งยินยอมให้มีการปรับปรุงเนื้อหาบนเว็บเพจได้โดยไม่ต้องโหลดหน้าเพจทั้งหมดใหม่ ซึ่งทำให้การรับส่งข้อมูลระหว่างเซิร์ฟเวอร์กับไคลเอนต์ต้องใช้ข้อความรูปแบบ JSON

2.2.8 JSON (Java Script Object Notation)

JSON [24] เป็นรูปแบบหนึ่งของการแลกเปลี่ยนข้อมูล ซึ่งเป็นภาษาที่มนุษย์และเครื่องจักรสามารถอ่านเข้าใจง่ายและสร้างขึ้นมาได้ เพราะอยู่ในรูปแบบของข้อความปกติ (plain text) ไม่จำกัด Data type ว่าต้องเป็นข้อความหรือตัวเลขเท่านั้น

การสร้าง JSON message มีด้วยกัน 2 โครงสร้าง คือ

- A collection of name/value pairs เสมือนการสร้าง Object ซึ่งจะมีชื่อและคุณสมบัติของ object นั้นๆ แนบมาด้วยกัน โดยจะแบ่งแต่ละ object ออกจากกันด้วยเครื่องหมายปีกกา { } และคั่น Name กับ Value ของ object ด้วยเครื่องหมาย colon (:) หรือ comma (,) เช่น

```
TweetA = {id: ‘1234567890’,
          screen_name: ‘userTest’,
          text: ‘Hello world’ }
```

- An ordered list of values เทียบเท่ากับการสร้าง Array ในภาษาอื่นๆ ซึ่งจะขึ้นต้นด้วยเครื่องหมาย [และลงท้ายด้วยเครื่องหมาย] เป็นการจบ list และใช้ comma (,) ในการแบ่ง value ใน list เช่น

  Tweets = [tweetA, tweetB, tweetC]

2.2.9 Indexed Database API

Indexed DB [25] นี้เป็น API หนึ่งสำหรับสร้างฐานข้อมูลออฟไลน์บนฝั่งไคลเอนต์ซึ่งช่วยให้การทำงานของแอพพลิเคชั่นง่ายมากขึ้นเมื่อต้องการจัดเก็บข้อมูลประเภท structured data ขนาดใหญ่สำหรับประมวลผลบนเบราว์เซอร์ โดยการเรียกใช้ API จะแบ่งการเข้าถึงเป็น 2 แบบให้เลือกคือ Asynchronous และ Synchronous แต่สำหรับแบบ Synchronous ค่อนข้างถูกนำมาใช้น้อยมาก มักจะใช้สำหรับ Web workers เท่านั้น ส่วน Asynchronous จะเป็นที่นิยมทั่วไป โดย Asynchornous API ที่ใช้เพื่อเข้าถึงฐานข้อมูลคือ open() จะได้รับข้อมูล IDBRequest object กลับมา ซึ่งการประมวลผลใดๆ จะถูก operate บน object นี้ ฐานข้อมูลที่สร้างขึ้นหรือเข้าถึงอยู่สามารถสร้าง Object Storage ขึ้นมาได้ เทียบได้กับ table ในฐานข้อมูลเชิงสัมพันธ์ และเพื่อกระทำการใดๆ กับข้อมูลนั้นจะต้องเปิด Transaction เพื่อเข้าถึงชุดข้อมูลก่อน โดยสามารถเปิดได้ 3 โหมดด้วยกัน คือ readonly, readwrite และ versionchange ดังตัวอย่างในรูปที่ 2.11

```
function Buffers() {
  var dbName = "buffers";
  var tableStatuses = "statuses";
  var dbObj;
  var store;
  var tx;
  var abortStatus = false;

  IDBCursor.PREV = IDBCursor.PREV || "prev";
  IDBCursor.NEXT = IDBCursor.NEXT || "next";

  return {
    init: function() {
      var openRequest = indexedDB.open(dbName, 10);
      //indexedDB.deleteDatabase(dbName);
      openRequest.onerror = function(e) {
        console.log("Buffer error: " + e.target.errorCode);
        indexedDB.deleteDatabase(dbName);
      };
      openRequest.onsuccess = function(e) {
        dbObj = openRequest.result;
        console.log("Buffer: created");
      };
      openRequest.onupgradeneeded = function (e) {
        if(e.currentTarget.result.objectStoreNames.contains("statuses")) {
          e.currentTarget.result.deleteObjectStore("statuses");
        }
        var objectStore = e.currentTarget.result.createObjectStore("statuses", {keyPath: "id"});
        objectStore.createIndex("keywordIndex", "keyword", { unique: false });
        console.log("Buffer: upgraded");
      };
    },

   openTransaction: function() {
     tx = dbObj.transaction(tableStatuses, "readwrite");
     store = tx.objectStore(tableStatuses);
   },
```

ภาพที่ 2.11 ตัวอย่างการเปิดใช้งาน Indexed DB

2.3 งานวิจัยที่เกี่ยวข้อง

เนื่องด้วยความสามารถในการเรียกค้นทวีตที่ทวิตเตอร์จัดเตรียมไว้ให้ ทำให้นักวิจัยมากมายสนใจและเข้าค้นคว้าเพื่อหาหนทางในการดึงสาระสำคัญที่เกิดขึ้นภายในเครือข่ายสังคมของผู้ใช้ทวิตเตอร์มาเป็นระยะเวลาหนึ่ง และต่อมาทวิตเตอร์ได้เพิ่มความสามารถในการระบุตำแหน่งเข้าไป เพื่อให้ผู้ใช้สามารถแนบพิกัดที่ตนอยู่ลงไปในทวีตหรือข้อความแต่ละครั้งได้อีกด้วย ดังนั้นงานวิจัยที่ออกมาจึงสามารถแบ่งออกได้เป็น 2 แบบ คือ การวิเคราะห์ข้อมูลเชิงเวลา และเชิงพื้นที่

2.3.1 งานวิจัยที่เกี่ยวข้องในการวิเคราะห์ข้อมูลจากทวิตเตอร์เชิงเวลา

การวิเคราะห์ข้อมูลบนสื่อสังคมออนไลน์ได้รับความสนใจมากขึ้นตามการขยายตัวและความนิยมของสื่อสังคมออนไลน์ ที่ผู้ใช้งานต่างให้ความคิดเห็นเกี่ยวกับเรื่องรอบตัวบนสื่อเหล่านี้ทำให้บริษัทที่ให้บริการและจำหน่ายสินค้าต่างๆ ต้องสำรวจข้อมูลเกี่ยวกับตนเองและคู่แข่งจากสื่อสังคมออนไลน์อยู่เสมอ เมื่อผนวกกับชุดคำสั่งที่เปิดกว้างของทวิตเตอร์ซึ่งช่วยให้นักพัฒนาสามารถดึงข้อมูลสตรีมมิ่ง (Streaming data) จากทวิตเตอร์ได้อิสระตามเงื่อนไขของชุดคำสั่งนั้นๆ ทวิตเตอร์จึงถูกนำไปใช้

เสมือนเป็น Crowd-sourced sensing [26] หรือเหมืองข้อมูล อย่าง Son Doan และคณะ [27] ที่ได้เสนอการติดตามโรคไข้หวัดใหญ่ผ่านทางทวิตเตอร์โดยอาศัยคีย์เวิร์ดที่เกี่ยวข้องกับโรคไข้หวัดใหญ่จาก BioCaster Ontology แล้วกรองข้อมูลด้วยหลักการ semantic อีกครั้งหนึ่งเพื่อแยกแยะด้วยอารมณ์ความรู้สึกเพิ่มเติม Hotstream [28] เป็นอีกหนึ่งแอพพลิเคชั่นที่ใช้ในการติดตามข่าวสารที่ทันต่อเหตุการณ์โดยรวบรวมแล้วจัดกลุ่มทำการเรียงลำดับด้วยตัวแปรความนิยมและความน่าเชื่อถือ ซึ่งช่วยให้ผู้ใช้สามารถติดตามข่าวสารนั้นๆ ได้ทันสมัย

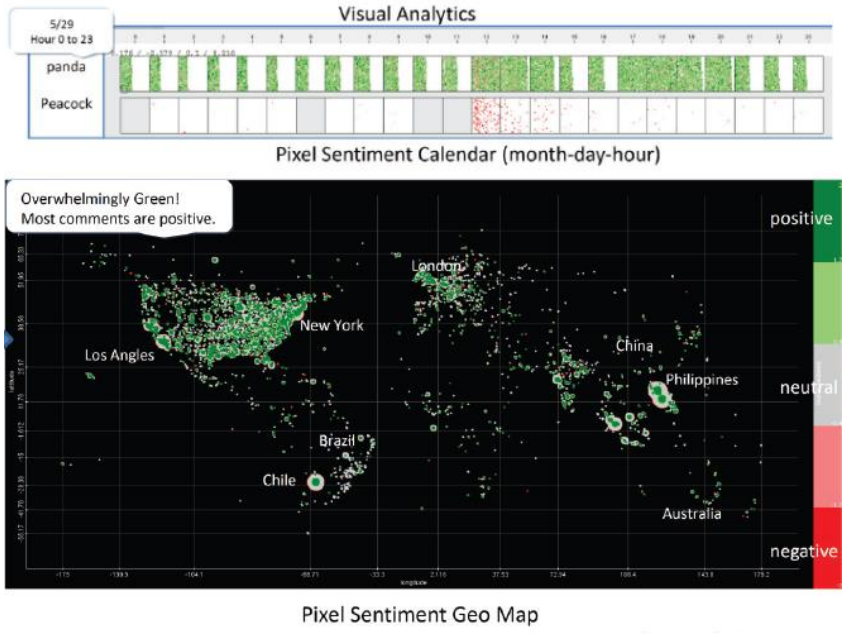
ทวิตเตอร์ถือเป็นช่องทางการสื่อสารที่นิยมที่สุดในบางประเทศ ดังนั้นจึงมีการประยุกต์ใช้เพื่อเตือนภัยทางธรรมชาติอย่างภัยสึนามิด้วยระบบ Twitter Early Warning ซึ่งถูกพัฒนาและนำมาทดลองใช้ในประเทศอินโดนีเซีย [29] ทวิตเตอร์ยังถูกทดลองใช้ในการทำนายผลการเลือกตั้งในประเทศสเปนเมื่อปีค.ศ. 2012 โดยใช้เครื่องมือชื่อ Taratweet ซึ่งนับจำนวนทวีตที่เกี่ยวกับพรรคการเมือง [4] อย่างไรก็ตาม ผลการทดลองค่อนข้างแตกต่างจากผลการเลือกตั้งจริง ซึ่งสาเหตุหนึ่งก็คือระบบไม่สามารถแยกได้ว่าข้อความที่พูดถึงพรรคการเมืองนั้นๆ ถูกพูดถึงไปในทางที่ดีหรือไม่ดี การตรวจสอบว่าข้อความนั้นๆ กำลังสื่อถึงความรู้สึกหรืออารมณ์แบบใดจึงเป็นอีกเรื่องที่น่าสนใจ จึงมีงานวิจัยที่ได้แบ่งแยกอารมณ์ของข้อความออกเป็น 6 แบบ โดยการแยกแยะแบบขั้นลำดับและได้ทำการทดลองผ่านชุดข้อมูลจากการแข่งขันฟุตบอล Brazilian Soccer League [30]

2.3.2 งานวิจัยที่เกี่ยวข้องในการวิเคราะห์ข้อมูลจากทวิตเตอร์เชิงพื้นที่

ทวิตเตอร์ได้เพิ่มความสามารถในการระบุพิกัดละติจูดและลองจิจูดให้กับแต่ละข้อความในเดือนสิงหาคมปี 2009 [31] ทำให้สามารถนำมาวิเคราะห์ในเชิงพื้นที่ได้ เช่น งานวิจัยของ Sakaki และคณะที่ได้นำ Kalman filter ซึ่งเป็นการประมาณค่ากำลังสองเชิงเส้นมาช่วยกรองข่าวสารจากทวีตและประมาณค่าหาจุดกำเนิดของการเกิดสึนามิในญี่ปุ่น [5] แต่ถึงแม้ว่าบริการระบุตำแหน่งจะถูกนำมาใช้บ่อยในทุกวันนี้ แต่ผู้ใช้งานส่วนใหญ่ก็ยังไม่อนุญาตให้แสดงพิกัดของตนเองในทวีต เนื่องจากประเด็นเรื่องความเป็นส่วนตัว ซึ่งปัญหานี้ทำให้มีนักวิจัยหาวิธีระบุตำแหน่งพิกัดของผู้ใช้งานโดยอาศัยข้อมูลจากข้อความในทวีตและการสนทนากับผู้ใช้งานอื่น [32]

นอกจากนี้ Ming Hao และคณะ [3] ยังสนใจวิเคราะห์ข้อความทวีตทั้งในเชิงพื้นที่และเชิงเวลาเพื่อกลั่นกรองหาข้อมูลที่มีประโยชน์สูงสุดในเรื่องหนึ่งๆ ทั้งแสดงข้อมูลได้อย่างน่าสนใจอีกด้วย โดยหลักการสำคัญคือกลั่นกรองเฉพาะทวีตที่เกี่ยวข้องกับหัวข้อที่สนใจ แล้วนำมาทำ Stream analysis เพื่อบ่งชี้ให้เห็นถึงความหนาแน่น การไหลของข้อมูล ความคิดเห็นในแง่บวกแง่ลบ แล้วจึง

นำมาแสดงผลโดยใช้จุดสีแต้มลงไปในปฏิทินทำให้ผู้ใช้มองเห็นว่าวันใดที่มีผู้ใช้ทวิตเตอร์พูดถึงสินค้าหรือบริการของตนมากที่สุดและเป็นไปในแง่ดีหรือไม่ดีซึ่งจะถูกแยกแยะโดยสีของจุดนั้นๆ และเพื่อตอบโจทย์ของผู้ใช้มากขึ้นจึงมีการปักหมุดลงใน geomap เพิ่มเติมไปด้วย ดังรูปที่ 2.12 ทำให้ผู้ใช้มองเห็นการรวมตัวหรือการกระจายของความสนใจในตัวสินค้าและบริการว่าอยู่ในบริเวณใดๆบ้าง นับเป็นเครื่องมือที่ตอบสนองความต้องการของผู้ใช้งานได้แทบจะครอบคลุมทุกด้านทีเดียว



ภาพที่ 2.12 จุดสีแทนความพึงพอใจบนปฏิทินและ GeoMap

ถึงแม้ว่าการอนุญาตให้เข้าถึงข้อความทวีตบนทวิตเตอร์เซิร์ฟเวอร์ได้อย่างอิสระนั้นจะสร้างความสะดวกและเป็นการสนับสนุนแก่นักพัฒนาทั้งหลาย แต่ในแง่ของความปลอดภัยก็ถือว่าเป็นเรื่องที่ค่อนข้างอันตรายมากทีเดียว Weidemann และ Swift [33] ได้ศึกษาการวิเคราะห์ทวีตที่แนบพิกัดพื้นที่เพื่อแสดงให้เห็นว่าข้อมูลที่ได้รับมานั้นสามารถระบุพฤติกรรมหรือชีวิตประจำวันของผู้ใช้หนึ่งๆได้ รวมถึงพื้นที่ครอบคลุมการใช้ชีวิตของผู้ใช้นั้นๆอีกด้วย จะเห็นว่าเครื่องมือที่ได้ออกมาจากการพัฒนาดังกล่าวนั้นสามารถสร้างคุณประโยชน์ได้หลากหลายไม่ว่าจะเป็นเพื่อการศึกษาหรือเพื่อทำธุรกิจ ทว่าข้อมูลเหล่านี้ก็สามารถนำไปสู่การเกิดอาชญากรรมที่ชาญฉลาดมากขึ้นได้เช่นกัน เพราะสามารถรู้ข่าวสารและการเคลื่อนไหวของเป้าหมายในชีวิตประจำวันง่ายมากขึ้น

บทที่ 3
การออกแบบและพัฒนาแอพพลิเคชั่น

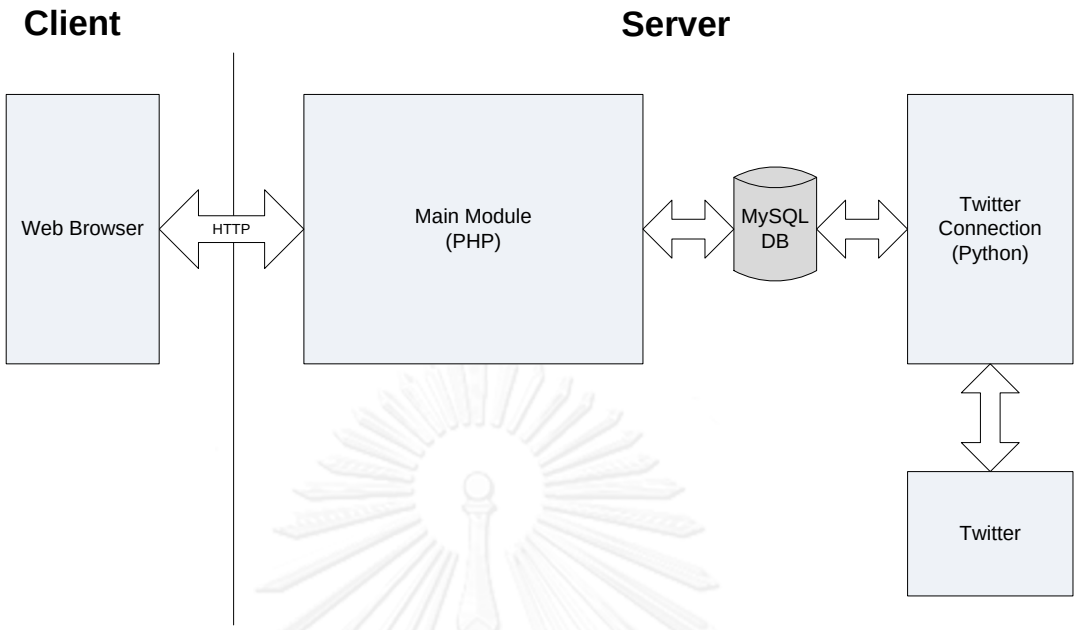
จากพื้นฐานโครงงาน [7] ระบบถูกออกแบบเป็นสองส่วนแบบไคลเอนต์เซิร์ฟเวอร์ ดังนั้นในงานวิจัยนี้ระบบ WOT (World of Tweeties) จะยังคงรูปแบบระบบเอาไว้ แต่เพิ่มเติมความสามารถบางอย่างเข้าไปให้กับระบบ เมื่อเปรียบเทียบกันจึงมีความแตกต่างดังนี้

- ในงาน [7] การเรียกคืนข้อความทวีต จะรับกลับมาได้เพียง 1-40 % ของข้อความทั้งหมดบนพื้นที่ทั่วโลกตามข้อจำกัดของ Streaming API ในขณะที่ระบบ WOT จะจำกัดขอบเขตการเรียกคืนให้เหลือเพียงประเทศไทยเพื่อให้ได้รับข้อความกลับมาในอัตราที่มากขึ้นเกือบ 100 %
- การแสดงผลจะเพิ่มส่วนของแผนภูมิเส้นและการคำนวณในเชิงตัวเลข นอกเหนือไปจากการแสดงหมุดข้อความบนแผนที่กูเกิลเอิร์ธ
- ระบบ WOT จะรองรับการค้นหาแบบหลายคำค้น เพื่อสนับสนุนการวิเคราะห์เปรียบเทียบข้อมูลระหว่างเรื่องที่สนใจ และสามารถกำหนดช่วงเวลาในการขอค้นได้
- เพิ่มเติมความสามารถในการจัดเก็บข้อความลงฐานข้อมูลถาวร (Archive) เพื่อเรียกดูผลลัพธ์ของการค้นหาในภายหลัง
- สนับสนุนการติดตามหัวข้อหนึ่งๆ โดยอนุญาตให้ผู้ใช้สามารถ Refresh การค้นหาแบบอัตโนมัติได้ทุกๆ 1 นาที, 5 นาที หรือ 10 นาที

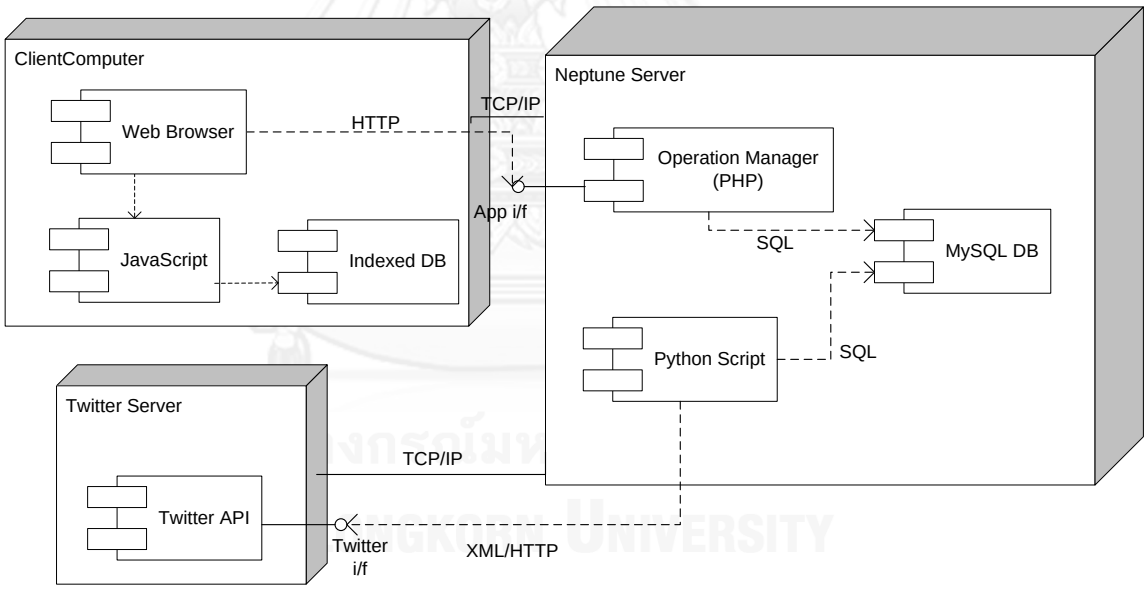
โดยที่มาของความสามารถดังกล่าวจะอธิบายรายละเอียดในหัวข้อย่อยถัดไป

3.1 ภาพรวมระบบ (Overview System)

งานวิจัยนี้ออกแบบระบบ WOT เป็น 2 ส่วน คือ ฝั่งไคลเอนต์ (Client side) และฝั่งเซิร์ฟเวอร์ (Server side) ดังรูปที่ 3.1 และ 3.2



ภาพที่ 3.1 ภาพรวมของระบบ WOT



ภาพที่ 3.2 สถาปัตยกรรมฮาร์ดแวร์และซอฟต์แวร์ของระบบ WOT

3.2 ฝั่งเซิร์ฟเวอร์ (Server Side)

WOT ถูกติดตั้งบนเซิร์ฟเวอร์ระบบปฏิบัติการ CentOS ซึ่งเป็นระบบลีนุกซ์แบบหนึ่ง ใช้ภาษา Python และ PHP ในการพัฒนา ทั้งนี้สามารถแบ่งงานออกเป็น 2 ส่วนหลัก คือ ส่วนการค้น

คืนทวีตจากทวิตเตอร์ และส่วนจัดการกลางของระบบ WOT โดยส่วนจัดการกลางระบบจะทำงานร่วมกันกับส่วนจัดการฐานข้อมูล

3.2.1 การค้นคืนทวีตจากทวิตเตอร์ (Tweet retrieval)

สำหรับการค้นคืนทวีตนี้จะถูกพัฒนาต่อยอดจากโครงงาน [7] ซึ่งพัฒนาด้วยภาษา Python เพื่อเพิ่มประสิทธิภาพในการคัดกรองและการจัดเก็บข้อมูลลงฐานข้อมูล MySQL แทน SQLite

Python script จะใช้ Tweepy เพื่อติดต่อกับทวิตเตอร์และเรียก Streaming API ในการร้องขอ Streaming tweets โดยการติดต่อทวิตเตอร์จะใช้วิธี OAuth ซึ่งต้องการลงทะเบียนในเว็บไซต์ https://dev.twitter.com/ เพื่อจะได้รับรหัสทั้งหมด 4 รหัส ได้แก่

- CONSUMER_KEY
- CONSUMER_SECRET
- ACCESS_TOKEN
- ACCESS_TOKEN_SECRET

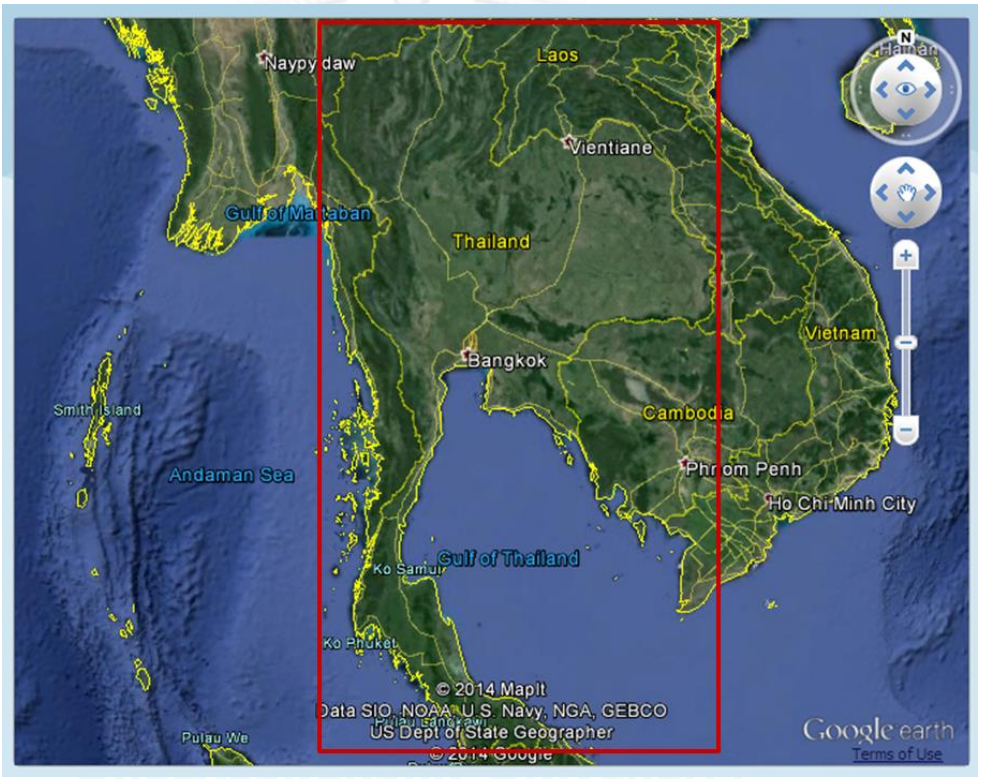
รหัสข้างต้นจะใช้เพื่อยืนยันตัวตนกับทวิตเตอร์เพื่อขอใช้บริการ ดังรูปที่ 3.3

```
def OAuth():

    auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
    auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
    tweepy.api = tweepy.API(auth)

    # Note: Had you wanted to perform the full OAuth dance instead of using
    # an access key and access secret, you could have uses the following
    # four lines of code instead of the previous line that manually set the
    # access token via auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
    #
    # auth_url = auth.get_authorization_url(signin_with_twitter=True)
    # webbrowser.open(auth_url)
    # verifier = raw_input('PIN: ').strip()
    # auth.get_access_token(verifier)

    return auth
```

ภาพที่ 3.3 OAuth function

แล้วจึงสามารถใช้งาน Streaming API เพื่อเรียกคืนทวีตได้ ซึ่งมีส่วนสำคัญคือ class CustomStreamListener(tweepy.StreamListener) โดย StreamListener นี้มีหลาย method ให้ใช้งาน และที่น่าสนใจหนึ่งในนั้นก็คือ on_status เป็น method เพื่อร้องขอทวีตที่เกิดขึ้นใหม่หรือก็คือรอรับทวีตแบบ real-time นั่นเอง

Stream object ที่ถูกสร้างขึ้นก็มี method ให้เรียกใช้อีกมากมาย ในที่นี้ filter() จะถูกหยิบยกมาใช้งาน และหนึ่งใน parameter ที่สามารถส่งไปเป็นเงื่อนไขในการขอรับทวีตนั้น ก็คือ location ที่กำหนดขอบเขตด้วย BBOX อันประกอบด้วยพิกัดมุมซ้ายล่าง (Bottom left: BL) และมุมขวาบน (Top right: TR) เรียงตามลำดับ [lonBL, latBL, lonTR, latTR] ตัวอย่างเช่น BBOX = [97.345619, 5.610000, 105.639381, 20.463181] ทั้งนี้จะเห็นได้จากรูปที่ 3.4 ว่าขอบเขต BBOX ที่ใช้นั้นนอกจากครอบคลุมพื้นที่ประเทศไทยแล้วยังกินพื้นที่บางส่วนของประเทศเพื่อนบ้านด้วย ดังนั้นข้อความทวีตทั้งหมดที่มี Geolocation ในบริเวณดังกล่าวจะถูกเรียกคืนมายังระบบ WOT ด้วย ตัวอย่างการระบุ locations เพื่อขอ streaming tweet เป็นดังรูปที่ 3.5



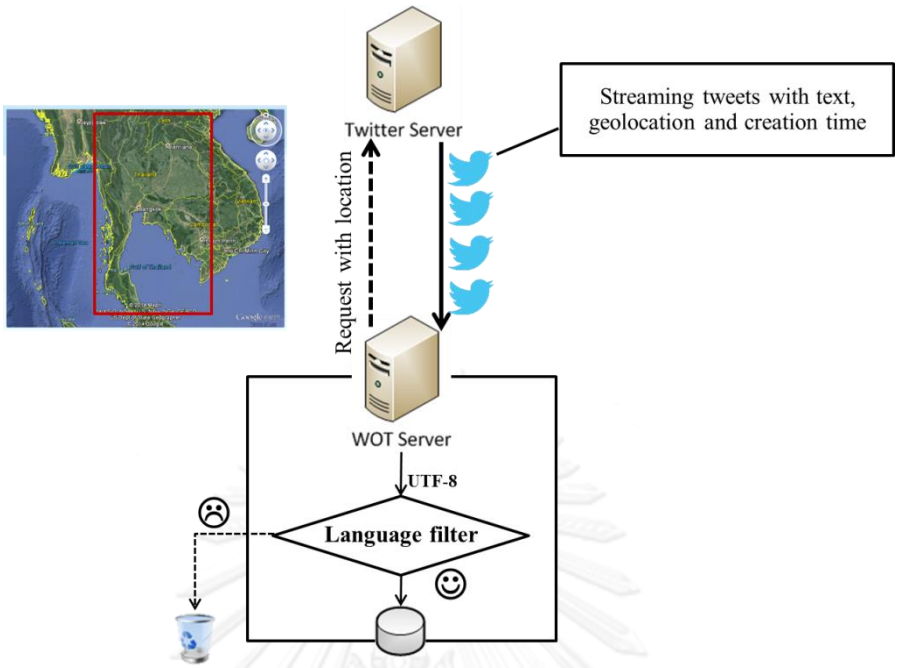
ภาพที่ 3.4 ขอบเขตประเทศไทยที่ใช้ในการเรียกคืนข้อความทวีต

```
def main():

       status_wrapper = TextWrapper(width=60, initial_indent='    ',
subsequent_indent='    ')
    auth = OAuth()
    while True:
        try:
                print "\n\n\n\n # Get streaming  msg ----------\n"
                # Create a streaming API and set a timeout value of 60 seconds
                streaming = tweepy.streaming.Stream(auth, CustomStreamListener(),
timeout=60)
                #Filtering with location
                streaming.filter(follow=None, track=None, async=False,
locations=[97.345619, 5.610000, 105.639381, 20.463181], count=None)
        except Exception, e :
                print "Main Error: %s" %(e)
if __name__ == '__main__': main()
```

ภาพที่ 3.5 Filtering stream by location

เมื่อทวีตเริ่มไหลเข้ามาตามท่อที่เปิดเอาไว้แล้ว ระบบจะจัดส่งแต่ละทวีตมากลั่นกรองด้วย Language filter หรือตัวคัดกรองภาษาก่อน ซึ่งจะคัดทวีตออกหากมีคำที่ไม่เหมาะสมอยู่ในทวีต หรือมีขนาดข้อความสั้นเกินกว่าจะมีความหมายที่น่าสนใจอยู่ ในที่นี้กำหนดไว้ 14 ตัวอักษรหรือก็คือ 10% ของความยาวทั้งหมดที่เป็นไปได้ ทวีตที่ตกเงื่อนไขดังกล่าวจะถูกทิ้งไปและรอรับทวีตใหม่เข้ามา ในทางตรงข้ามทวีตที่ผ่านการคัดกรองจะถูกนำส่งเข้าฐานข้อมูล MySQL เก็บเอาไว้ ดังรูปที่ 3.6

การจัดเก็บข้อความทวีตจะจัดลงฐานข้อมูล MySQL v5.1 ซึ่งเป็นฐานข้อมูลเชิงสัมพันธ์ (Relational Database) จะเก็บข้อมูลในรูปของตารางมีลักษณะ 2 มิติ คือ แถว กับ คอลัมน์ โดยในที่นี้จะสร้างตารางชื่อ statuses เพื่อจัดเก็บข้อมูลทวีตที่ได้รับ ประกอบด้วยคอลัมน์ดังในตารางที่ 3.1 และจัดเก็บทวีตลงฐานข้อมูลดังรูปที่ 3.7



ภาพที่ 3.6 ภาพรวมของระบบฝั่ง Server ส่วนเรียกคืนทวีต

ตารางที่ 3.1 โครงสร้างของฐานข้อมูลตาราง Statuses

ชื่อคอลัมน์	**คำจำกัดความ**
ID	รหัสของข้อความทวีต
TEXT	ข้อความทวีต
SOURCE	อุปกรณ์หรือแหล่งที่มาของข้อความทวีต
CREATED_AT	วันเวลาที่ผู้ใช้ทวิตเตอร์ทวีตข้อความขึ้นมา
LAT	ละติจูดแสดงพิกัดของข้อความ
LON	ลองจิจูดแสดงพิกัดของข้อความ
USERID	รหัสผู้ใช้งาน
SCREEN_NAME	นามแฝงของเจ้าของข้อความบนทวิตเตอร์
PROFILE_IMAGE_URL	ลิงค์รูปถ่ายของผู้ใช้ทวิตเตอร์
RETWEETED	เป็นข้อความที่ถูกรีทวีตหรือไม่
RETWEET_COUNT	จำนวนที่ถูกรีทวีต

```
#!/usr/bin/python
##-*-coding: utf-8 -*-


conn =
mdb.connect(host=db_server,user=db_uname,passwd=db_pass,db=db_name,use_unicode=Tr
ue,charset="utf8")

# Create the MySQL cursor that is used to query database
queyrCurs = conn.cursor()

def
addStatus(id,text,source,created_at,lat,lon,userid,screen_name,profile_image_url,
retweeted,retweet_count):
    try:
        # Calls the execute method that will submit a insert SQL Query
        sql = "INSERT INTO statuses
(id,text,source,created_at,lat,lon,userid,screen_name,profile_image_url,retweeted
,retweet_count) VALUES (%d, '%s', '%s','%s', '%s','%s', %d, '%s', '%s', %s,
%s);"%(id,mdb.escape_string(text),mdb.escape_string(source),mdb.escape_string(cre
ated_at),mdb.escape_string(lat),mdb.escape_string(lon),userid,mdb.escape_string(s
creen_name),mdb.escape_string(profile_image_url),retweeted,retweet_count)
        queryCurs.execute(sql)
        print("Insert Success!")

    except Exception, e:
        print "Error to insert DB: %s" %e
```

ภาพที่ 3.7 การ Insert Tweet ลง MySQL DB

3.2.2 ส่วนจัดการฝั่งเซิร์ฟเวอร์ (Operation Manager)

Operation manager มีหน้าที่ประสานงานของมอดูลทางฝั่งเซิร์ฟเวอร์เพื่อตอบสนองต่อคำร้องจากฝั่งไคลเอนต์โดยจัดส่งข้อมูลที่เหมาะสมกลับไปให้ ซึ่งคำร้องที่ระบบสามารถรองรับได้เป็นไปตามตารางที่ 3.2 จะเห็นว่าคำร้องสามารถแบ่งออกเป็น 2 หัวข้อหลัก คือ คำร้องที่เกี่ยวข้องกับการเรียกค้นทวีตในฐานข้อมูล และคำร้องที่เกี่ยวกับการจัดการหน้าเพจ

ตารางที่ 3.2 รูปแบบคำร้องขอจากไคลเอนต์

URL	Function การทำงาน
"ajax.php?all_tweet=true&bbox="+bbox+"&startdate="+startDate+"&enddate="+endDate	searchTweetWithBBox: ระบบจะไป Query หรือเรียกค้นทวีตทั้งหมดที่อยู่ในขอบเขตของพื้นที่ BBOX และช่วงเวลาระหว่าง StartDate กับ EndDate
"ajax.php?keyword="+encodeURIComponent(keyword)+"&bbox="+bbox+"&startdate="+startDate+"&enddate="+endDate+"&searchmode="+searchMode;	searchTweet : ทำการ Query ทวีตด้วยคีย์เวิร์ดที่ผู้ใช้งานระบุมาและกรองเฉพาะทวีตที่อยู่ในพื้นที่ BBOX และช่วงเวลาระหว่าง StartDate กับ EndDate

URL	Function การทำงาน
"ajax.php?all_archive=true"	loadAllArchiveList: ทำการ list keword ทั้งหมดที่ save อยู่ในตาราง archive list
"ajax.php?load_archive="+archiveID	loadArchive: Query ข้อความทวีตของ archive ID นั้นๆซึ่งถูกจัดเก็บแยกจากข้อมูลหลัก อยู่ในตาราง archive data
"ajax.php?delete_archive="+archiveID	deleteArchive: ลบข้อมูลที่อยู่ใน archive list และ archive data
"ajax.php?load_config=true"	loadConfig: Query ค่า config จากตาราง setting
"ajax.php?save_config=true&tweetperload="+$("#labelTweetLoad").html()+"numberkeyword="+$("#labelNumberKeyword").html()	saveConfig: บันทึกค่า TweetPerLoad กับ NumberOfKeyword ที่ได้จากการตั้งค่าผ่านหน้าเว็บเพจลงตาราง setting

ทั้งนี้พารามิเตอร์ bbox ที่ถูกสร้างขึ้นจากเบราว์เซอร์ฝั่งไคลเอนต์นั้นจะประกอบด้วยค่าละติจูดและลองจิจูดของมุมมอง ณ ขณะนั้นของแผนที่กูเกิลเอิร์ธ ในลักษณะเรียงจาก ละติจูดฝั่งตะวันตก ละติจูดฝั่งตะวันออก ลองจิจูดฝั่งใต้ และลองจิจูดฝั่งเหนือ

3.2.3 ส่วนจัดการฐานข้อมูล (Database Manager)

ส่วนจัดการฐานข้อมูลจะทำหน้าที่ประสานงานระหว่างแอพพลิเคชั่นกับฐานข้อมูล โดยฟังก์ชันที่ Operation manager จัดหาให้แต่ละคำสั่งนั้นจะลิงค์มายังส่วนนี้เพื่อทำการ query หรือบันทึกข้อมูลลงในฐานข้อมูล “testbed” ซึ่งบรรจุตารางที่จำเป็นต่างๆ เอาไว้ให้ ตารางที่เกี่ยวข้องกับระบบได้แก่ statuses table, archive_list table, archive_data table และ setting โครงสร้างของแต่ละตารางจะเป็นไปตามรูปที่ 3.8 ถึง 3.11

	#	Name	Type
☐	1	id	bigint(40)
☐	2	text	text
☐	3	source	text
☐	4	created_at	datetime
☐	5	lat	text
☐	6	lon	text
☐	7	userid	int(11)
☐	8	screen_name	text
☐	9	profile_image_url	text
☐	10	retweeted	tinyint(1)
☐	11	retweet_count	int(7)

ภาพที่ 3.8 โครงสร้างตาราง statuses

	#	Name	Type
☐	1	id	int(4)
☐	2	keyword	text
☐	3	current_view	text
☐	4	bbox	text
☐	5	start_date	datetime
☐	6	end_date	datetime
☐	7	time_stamp	datetime
☐	8	search_mode	char(3)

ภาพที่ 3.9 โครงสร้างตาราง archive_list

	#	Name	Type
☐	1	archive_id	int(4)
☐	2	id	bigint(40)
☐	3	text	text
☐	4	source	text
☐	5	created_at	datetime
☐	6	lat	text
☐	7	lon	text
☐	8	userid	int(11)
☐	9	screen_name	text
☐	10	profile_image_url	text
☐	11	retweeted	tinyint(1)
☐	12	retweet_count	int(7)
☐	13	keyword	text

ภาพที่ 3.10 โครงสร้างตาราง archive_data

	#	Name	Type
☐	1	tweet_download_per_time	int(6)
☐	2	number_of_keyword	int(1)

ภาพที่ 3.11 โครงสร้างตาราง setting

3.3 ฝั่งไคลเอนต์ (Client Side)

เบราว์เซอร์ฝั่งไคลเอนต์จะช่วยลดภาระงานของฝั่งเซิร์ฟเวอร์ ซึ่งถูกโปรแกรมผ่าน JavaScript โดยฝั่งเซิร์ฟเวอร์จะส่งมาพร้อมกับหน้าเว็บเพจ JavaScript ที่ถูกแนบมาด้วยได้แก่

3.3.1 การจัดการฝั่งไคลเอนต์ (Client Operation)

ตัวจัดการฝั่งไคลเอนต์นี้มีหน้าที่เสมือนหน่วยจัดการกลางของระบบช่วยประสานงานระหว่างมอดูลต่างๆ ในระบบให้สามารถทำงานและตอบสนองความต้องการของผู้ใช้ได้อย่างเหมาะสม โดยประกอบไปด้วยฟังก์ชันการทำงานต่างๆ ดังนี้

- Search function: เมื่อผู้ใช้ทำการค้นหาหัวข้อที่สนใจผ่าน text box ที่จัดเตรียมไว้ให้ ตัวจัดการฝั่งไคลเอนต์จะถูกเรียกเพื่อตบแต่ง URL ไป Query ข้อมูลทางฝั่งเซิร์ฟเวอร์

- startAsyncLoad function: หลังจากร้องขอข้อมูลไปทางเซิร์ฟเวอร์ ระบบจะทำการจัดส่งข้อมูลกลับมาให้ในรูปแบบของ JSON message ซึ่งไคลเอนต์จะนำไปจัดเก็บลงใน index buffer ด้วย function นี้
- startTweetsProcessor: เรียก Google Earth module เพื่อทำการปักหมุดข้อความลงบนแผนที่
- onProcessFinish function: จะถูกเรียกเมื่อจบการทำงานของ module หนึ่งๆ เพื่อเตรียมเรียก module ถัดไป โดยการทำงานของระบบจะมีลำดับขั้นดังนี้
    - Tweet loading: โหลดสตรีมมิ่งทวีตจากเซิร์ฟเวอร์มาเก็บไว้ใน buffer
    - Google Earth render: ปักหมุดทวีตที่ได้รับลงบนแผนที่กูเกิลเอิร์ธ
    - Google chart: วาดแผนภูมิแสดงแนวโน้มในช่วงเวลาที่กำหนด
    - Tweet analytics: วิเคราะห์ข้อมูลเชิงพื้นที่จากกลุ่มทวีต
    - Tweet lists: แสดงข้อความของกลุ่มทวีตที่นำมาวิเคราะห์บนหน้าเว็บ

  ตัวอย่างการใช้งานคือ เมื่อ Tweet loading ทำงานเสร็จสิ้นแล้ว Google Earth render จะถูกเรียกเพื่อให้ทำงานต่อผ่านฟังก์ชันนี้
- onViewChange function: จัดเก็บ BBOX ใหม่เมื่อมีการเปลี่ยนมุมมองบนหน้าแผนที่กูเกิลเอิร์ธ หากอยู่ในระหว่างโหมดค้นหาและไม่ติดสถานะหยุดนิ่ง (Freeze) ระบบจะทำการค้นหาหัวข้อนั้นๆ บนพื้นที่ใหม่โดยกลับไปเรียกที่ Search function อีกครั้ง หากอยู่ในโหมด Archive จะทำการเรียก startAsyncLoad function แทนโดยระบุ ID และ BBOX ใหม่ลงไปใน URL ให้เหมาะสม
- Clear และ Abort function: เพื่อยกเลิกทุกการทำงานของระบบฝั่งไคลเอนต์
- onBodyLoad function: จัดเก็บค่าที่เกิดจากผู้ใช้งานปรับแต่งบนหน้าเว็บเพจลงบนเซิร์ฟเวอร์
- onMapTweetClick function: เมื่อผู้ใช้คลิกไปที่ทวีตบนแผนที่ระบบจะเลื่อนมุมมองไปหาทวีตนั้นๆ และขยายภาพบนแผนที่กูเกิลเอิร์ธ
- addTweetInfo function: จะถูก Tweet Analytics module เรียกเมื่อมอดูลทำการประมวลกลุ่มข้อความจนได้ข้อมูลครบถ้วนแล้ว เพื่อแสดงผลบนหน้าเพจ
- onFurthestDistanceOfPropagationListener function: จะไปเรียก Google Earth module เพื่อวาดเส้นเชื่อมระหว่างข้อความทวีต 2 ข้อความ

จากที่กล่าวมาจะเกี่ยวกับการทำงานอย่างเป็นลำดับ (Sequence) เพื่อสืบค้นข้อความทวีตที่สนใจจากฐานข้อมูลของระบบ ซึ่งจะไม่มีฟังก์ชันใดทำหน้าที่วิเคราะห์กลุ่มข้อมูลที่ได้รับมาโดยตรง แต่จะช่วยควบคุมและรับส่งข้อมูลในการทำงานของระบบให้เป็นไปอย่างเหมาะสม รวมถึงหน้าที่การจัดวางข้อมูลหลังวิเคราะห์บนหน้าเว็บเพจด้วย

- Home function: เพื่อเคลียร์ข้อมูลใน buffer และหยุดกิจการต่างๆ ที่กำลังทำอยู่เพื่อกลับไปสู่หน้าเริ่มต้น
- toggleFreezeUpdate function: ตั้งสถานะหยุดชั่วคราวเพื่อป้องกันการโหลดข้อมูลใหม่เมื่อเปลี่ยนมุมมอง
- showChart และ showInfo: เนื่องด้วยหน้าเว็บเพจมีพื้นที่จำกัดในการใช้สอย ผู้พัฒนาจึงทำการซ่อนเนื้อหาบางส่วนเอาไว้ เมื่อทำการคลิกลูกศรที่ปรากฏอยู่ดังรูปที่ 3.12 ระบบจะเรียกฟังก์ชันนี้เพื่อแสดงเนื้อความของส่วนนั้นขึ้นมา
- lookAtTweetOnMap function: ถูกใช้เมื่อต้องการเลื่อนหน้าจอของแผนที่ไปที่ตำแหน่งหนึ่งๆ
- setRefreshPage function: เพื่อสนับสนุนความสามารถในการมอนิเตอร์ข้อมูลโดยการ refresh การค้นหาแบบอัตโนมัติ ซึ่งมีทั้งหมด 4 โหมดให้เลือก คือ No refresh หรือ Refresh ทุก 1 นาที, 5 นาที และ 10 นาที

กลุ่มฟังก์ชันย่อยที่กล่าวมาถูกจัดทำเพื่ออำนวยความสะดวกแก่ผู้ใช้งานแอพพลิเคชั่น นอกจากนี้ยังมีส่วนของ Archive สำหรับจัดเก็บกลุ่มทวีตที่น่าสนใจไว้ในฐานข้อมูลถาวรอีกด้วย และเพื่อสนับสนุนการทำงานของ Archive นี้ ในตัวจัดการอย่าง Client operation จึงต้องมีฟังก์ชันต่อไปนี้

- archiveData function: จะถูกเรียกใช้เมื่อผู้ใช้งานเลือกจัดเก็บสิ่งที่ค้นหาลงฐานข้อมูลถาวร โดยเบราว์เซอร์จะจัดส่งข้อมูลที่จำเป็นไปให้เซิร์ฟเวอร์เพื่อเป็นคีย์ในการจัดเก็บทวีตต้นฉบับต่อไป ได้แก่ คีย์เวิร์ดที่ใช้ค้นหา, BBOX, startDate, endDate และ search mode (ค้นหาโดยชื่อผู้ใช้ทวิตเตอร์หรือค้นหาจากข้อความ)
- showArchiveList function: จัดส่ง URL เพื่อร้องขอ list keyword ทั้งหมดที่ถูกเก็บอยู่ใน archive เพื่อมาแสดงบนหน้าเพจ

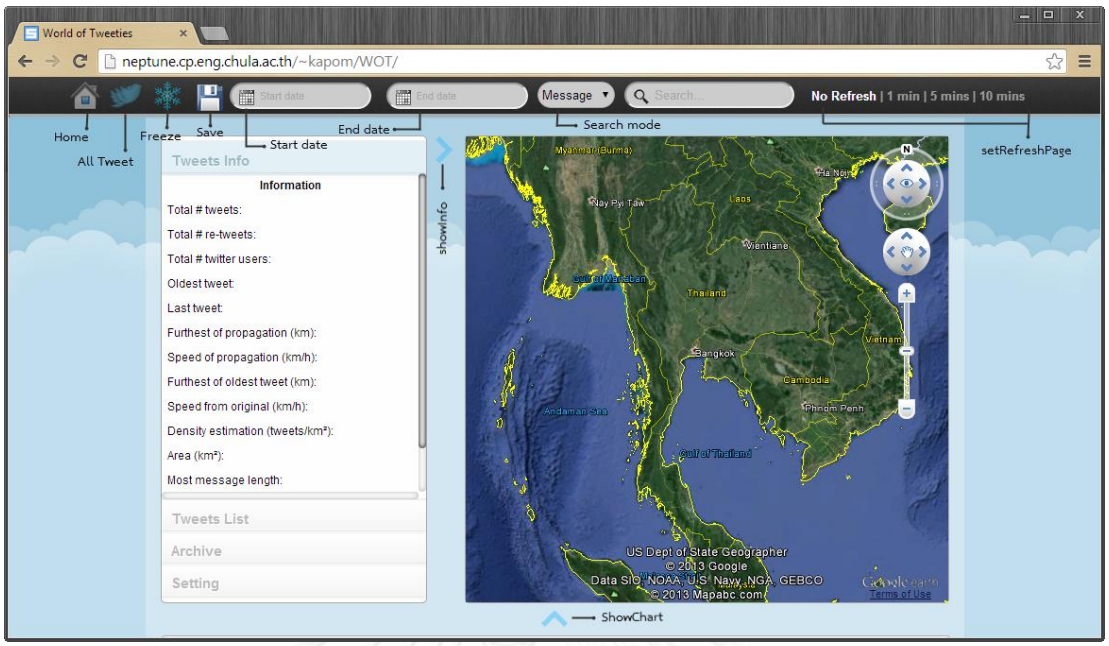
- loadArchive function: จะถูกเรียกใช้เมื่อผู้ใช้งานทำการคลิกเลือกดูข้อมูลเก่าจาก archive list ซึ่งฟังก์ชันนี้จะจัดส่ง URL ไปให้เซิร์ฟเวอร์ใช้ค้นคืนทวีตโดยอาศัย ID ที่แนบมาใน URL ในการค้นหา

และส่วนสุดท้ายเกี่ยวกับการตั้งค่าเบื้องต้นของระบบซึ่งจะมีด้วยกัน 2 ค่า คือ Tweet download และ Number of keywords โดย Tweet download คือค่าที่ใช้กำหนดจำนวนทวีตสูงสุดในการโหลดหนึ่งรอบ เนื่องจากข้อความทวีตในระบบมีปริมาณมากผู้พัฒนาจึงออกแบบให้การค้นคืนทวีตแบ่งทำเป็นรอบๆ ซึ่งในหนึ่งรอบจะถูกกำหนดจำนวนทวีตด้วยค่านี้ผ่านหน้าเว็บเพจ สำหรับ Number of keyword จะใช้เพื่อกำหนดคีย์เวิร์ดสูงสุดในการค้นหาแต่ละครั้ง

- loadConfig function: ร้องขอข้อมูลจากเซิร์ฟเวอร์เพื่อแสดงค่าที่ถูกตั้งไว้บนหน้าเพจ
- saveConfig function: เมื่อมีการเปลี่ยนแปลงค่าทั้งสองผ่านเว็บเพจ ตัวจัดการจะส่งข้อมูลที่เปลี่ยนแปลงไปยังเซิร์ฟเวอร์เพื่อขอให้บันทึกลงในฐานข้อมูลแทนค่าเก่า

3.3.2 การจัดการแผนที่กูเกิลเอิร์ธ (Google Earth Manager)

Google Earth manager จะมีฟังก์ชันการทำงานหลักคือการสร้าง KML file เพื่อสร้าง Placemark ให้กับแต่ละข้อความบนแผนที่ วาดเส้นเชื่อมระหว่างทวีตที่ใกลที่สุดหรือเก่าที่สุดไปใหม่ที่สุด โดยโครงสร้างไฟล์ KML สำหรับสร้าง placemark จะเป็นไปตามรูปที่ 3.13 และสำหรับสร้างบอลลูนข้อความตามรูปที่ 3.14 ในส่วนของการวาดเส้นจะได้ KML file ตามรูปที่ 3.15



ภาพที่ 3.12 หน้าเว็บ WOT

```
kml='<?xml version="1.0" encoding="UTF-8"?><kml xmlns="http://www.opengis.net/kml/2.2"><Document>'
kml+='<Placemark id="'+id+'">'
kml+= self.getRetweetBalloon(tweet);
kml+='<Point>'+
        '<coordinates>'+tweet.lon+','+tweet.lat+',0</coordinates>'+
     '</Point>'+
     '<Style>'+
         '<IconStyle>'+
         '<Icon>'+
             '<href><![CDATA[http://neptune.cp.eng.chula.ac.th/~kapom/WOT/util/TweetIcon.php]]></href>'+
         '</Icon>'+
         '</IconStyle>'+
     '</Style>'+
    '</Placemark>'+
    '</Document></kml>';
```

ภาพที่ 3.13 โครงสร้าง KML file สำหรับข้อความทวีต

```
getTweetBalloon: function(tweet) {
    var balloon = '<description><![CDATA[<table width="350" height="80">'+
                    '<tr>'+
                        '<td width="20%" rowspan="2" align="center">'+
                            '<img src="'+tweet.profileImgURL+'" width="32" height="32">'+
                        '</td>'+
                        '<td width="80%">'+
                            '<table width="100%" border="0" cellspacing="0" cellpadding="0">'+
                                '<tbody>'+
                                    '<tr>'+
                                        '<td width="60%">'+
                                            '<strong>'+tweet.screenName+'</strong>'+
                                        '</td>'+
                                        '<td width="40%" align="center">'+self.toHumanTime(tweet.createAt)+'</td>'+
                                    '</tr>'+
                                '</tbody>'+
                            '</table>'+
                        '</td>'+
                    '</tr>'+
                    '<tr>'+
                        '<td align="left" valign="top">'+self.highlightWords(tweet.text, tweet.keyword)+
                        '<br>'+
                        'Geo: '+tweet.lat+","+tweet.lon+
                        '</td>'+
                    '</tr>'+
                '</table>]]></description>';
    return balloon;
},
```

ภาพที่ 3.14 โครงสร้าง Description สำหรับสร้างบอลลูนข้อความใน Placemark

```
drawLine: function(tweetA, tweetB, keyword, keywordIndex, lineWidth) {
    var keyword = keyword.replace(/"/g, "");
    var kmlLine='<?xml version="1.0" encoding="UTF-8"?>\n'+
            '<kml xmlns="http://www.opengis.net/kml/2.2"><Document>\n'+
                '<Placemark id="'+keyword+'">\n'+
                 '<visibility>0</visibility>'+
                 '<LineString>\n'+
                  '<extrude>1</extrude>\n'+
                  '<tessellate>1</tessellate>\n'+
                  '<altitudeMode>clampToGround</altitudeMode>\n'+
                  '<coordinates>\n'+
                   tweetA.lon +','+ tweetA.lat +',0 \n'+
                   tweetB.lon +','+ tweetB.lat +',0 \n'+
                  '</coordinates>\n'+
                 '</LineString>\n'+

                 '<Style>'+
                  '<LineStyle>'+
                   '<color>'+colorList[keywordIndex]+'</color>'+
                   '<width>'+lineWidth+'</width>'+
                   '</LineStyle>'+
                 '</Style>'+
                '</Placemark>'+
                '</Document>'+
            '</kml>';
    var kmlObject = ge.parseKml(kmlLine);
    ge.getFeatures().appendChild(kmlObject);
},
```

ภาพที่ 3.15 โครงสร้าง KML file เพื่อวาดเส้นเชื่อมระหว่างทวีต

3.3.3 การจัดการแผนภูมิ (Chart Manager)

Chart manager เป็นมอดูลหนึ่งที่ใช้เพื่อวาดแผนภูมิเทียบจำนวนทวีตที่เกี่ยวพันกับหัวข้อที่ใช้ค้นหากับช่วงเวลาหนึ่งๆ ซึ่งสามารถกำหนดแกนเวลาได้สองแบบคือ รายวัน และ รายชั่วโมง โดยมอดูลเลือกแกนใดขึ้นอยู่กับการเลือกแบบการค้นหา หากเลือกการค้นหาโดยไม่สนใจช่วงวันที่หรือ

เวลา มอดูลนี้จะเลือกแสดงแกนเวลาแบบเป็นรายวันยาว 7 วัน แต่ในทางกลับกันหากส่งช่วงวันหรือเวลามาให้มอดูลจะจัดแสดงแกนเวลาเป็นรายชั่วโมงแทน

ในส่วนแรก buildChartData function จะจัดเตรียมข้อมูลเพื่อทำการวาดแผนภูมิไว้ก่อน ซึ่งก็คือการระบุคอลัมน์ที่ต้องการเปรียบเทียบลงไป จากนั้นจึงนำข้อมูลที่ได้รับมาจากตัวจัดการมาเพิ่มเข้าไปในตารางให้ครบถ้วน แล้วจึงวาดแผนภูมิบนหน้าเว็บด้วย chart.draw() ซึ่งมีมาให้กับไลบรารี่ของกูเกิล และวนกระทำแบบนี้จนครบตามจำนวนคีย์เวิร์ดที่ใส่เข้ามา ในส่วนของ buildChartData จะแสดงโค้ดได้ดังรูปที่ 3.16 สำหรับการจัดคอลัมน์และใส่ข้อมูลแต่ละแถว ผลลัพธ์จากโค้ดดังกล่าวจะได้หน้าตาออกมาดังรูปที่ 3.17

3.3.4 การวิเคราะห์ทวีต (Tweet Analytics)

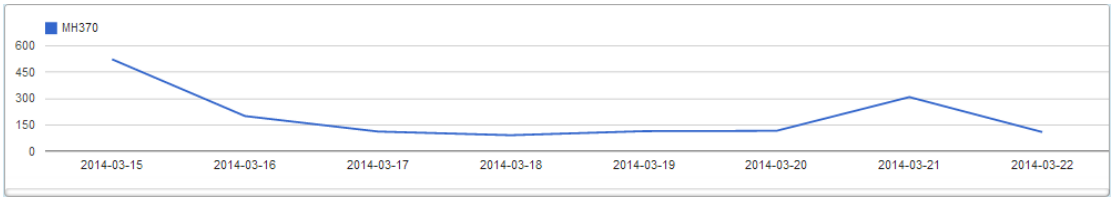
การวิเคราะห์ทวีตในที่นี้จะนำเสนอข้อมูลเชิงพื้นที่บนหน้าเพจ ดังเช่น ระยะทางที่ไกลที่สุดที่เกิดขึ้น ความเร็วในการแพร่กระจาย หรือความหนาแน่นของทวีตในพื้นที่

เริ่มต้นด้วยการค้นหาทวีตที่เก่าที่สุด และทวีตที่อยู่บนขอบนอกที่สุดของทางเหนือ ใต้ ตะวันตก ตะวันออก ตะวันตกเฉียงเหนือและเฉียงใต้ ตะวันออกเฉียงเหนือและเฉียงใต้ รวมเป็น 8 ข้อความ ทั้งนี้เพราะโอกาสที่ข้อความจะมีพิกัดเดียวกันได้นั้นมีน้อยมากจึงแทบเป็นไปไม่ได้ที่จะมีข้อความเกิดบนแนวละติจูดหรือลองจิจูดเดียวกัน โดยการคัดเลือกทวีตทั้ง 8 ข้อความนั้นได้จากการคำนวณระยะทางแบบยูคลิดระหว่างข้อความนั้นเทียบกับขอบของกรอบกูเกิลเอิร์ธซึ่งถูกระบุไว้ด้วยละติจูดและลองจิจูด ดังรูปที่ 3.18 ทำให้ได้โปรแกรมตามรูปที่ 3.19 เมื่อได้ข้อความที่ต้องใช้แล้วจึงตรวจสอบสถิติความยาวของข้อความทวีตทั้งหมดและแหล่งที่มาของทวีตนั้นๆว่าใช้อุปกรณ์ใดบ้างเพื่อเก็บเป็นข้อมูลเบื้องต้น

การวัดระยะทางที่ไกลที่สุดจะใช้ 8 ข้อความที่หาได้ในส่วนต้นมาทำการวัดระยะด้วย getDistance function ซึ่งอาศัยหลักของ Haversine ดังรูปที่ 3.20 เพื่อคำนวณระยะทางระหว่างพิกัด แล้วเปรียบเทียบระยะทางที่เกิดขึ้นจาก 8x7 คู่ ส่วนระยะทางจากทวีตที่เก่าที่สุดไปหาทวีตที่ไกลที่สุดนั้นจะใช้หลักการเดียวกันเพียงแต่จะวัดจากจุดตั้งต้นเดียวคือทวีตที่เก่าที่สุด ทำให้ได้ระยะทางที่ต้องนำมาเทียบกันทั้งหมด 1x8 คู่ สุดท้ายจะได้เป็นระยะทางระหว่างทวีตที่ไกลที่สุด กับระยะทางของทวีตที่เก่าที่สุดไปไกลที่สุด ทั้งนี้ระยะทางที่ไกลที่สุดที่เกิดขึ้นจะใช้เป็นตัววัดความกว้างของขอบข้อมูล ส่วนระยะทางจากจุดกำเนิดไปหาทวีตที่อยู่ไกลที่สุดนั้นจะเป็นตัววัดการกระจายของข้อมูลว่าจากจุดเริ่มแรกที่มีคนพูดถึงนั้นข้อมูลถูกพูดต่อหรือคุยเรื่องเดียวกันนี้ไกลแค่ไหน

```
buildChartData: function(startTime, endTime, keyword, callbackFunction) {
        if(abortStatus) return;

        var stepUp = 86400; // 1 day
        if(hourlyMode) {
                stepUp = 3600; // 1 hour
        }
        tmpTime =  new Date(endTime * 1000);
        endTime = new Date(tmpTime.getFullYear(), tmpTime.getMonth(),

tmpTime.getDate(), 23, 59).getTime();
        endTime = endTime / 1000;
        startTime = endTime - 691140;  // 7 days ,23 hours and 59 mins
        chartData = new google.visualization.DataTable();

        // Add Date time column
        chartData.addColumn('string', 'Date time');

        // Add keyword columns
        for (var i = 0; i < keyword.length; i++) {
                chartData.addColumn('number', keyword[i]);
        }
        // Initial chartData
        for(j=startTime; j<endTime; j+=stepUp) {
                data = [''];
                for(i=0; i<keyword.length; i++) {
                        data.push(0);
                }
                chartData.addRow(data);
        }
        var row = 0;
        for(i=startTime; i<endTime; i+=stepUp) {
                if(abortStatus) return;
                var date = new Date(i * 1000);
                var dateColumnData = '';
                if(hourlyMode) {
                        dateColumnData = date.getFullYear()+"-"+
                                         self.padDigits(date.getMonth()+1,2)+
                                         "-"+self.padDigits(date.getDate(),2)+" "+
                                         self.padDigits(date.getHours(),2)+":"+
                                         self.padDigits(date.getMinutes(),2);
                }
                else {
                        dateColumnData = date.getFullYear()+"-"+
                                          self.padDigits(date.getMonth()+1,2)+
                                          "-"+self.padDigits(date.getDate(),2);
                }

                chartData.setValue(row, 0, dateColumnData);

                for(j=0; j<keyword.length; j++) {
                        if(abortStatus) return;

                        var column = j + 1;

        infoCountStack.push(keyword[j]+","+i+","+(i+stepUp)+","+row+","+column);
                }
                row ++;
        }
        self.startSyncronizeCount(0); // Start syncronize counter
},
```

ภาพที่ 3.16 การเตรียม Chart data เพื่อสร้างแผนภูมิ



ภาพที่ 3.17 ตัวอย่างแผนภูมิที่ได้จากระบบ WOT

ภาพที่ 3.18 ตัวอย่างการกำหนดขอบบนหน้าต่างกูเกิลเอิร์ธ

```
// Keep tweet edge of bbox
if(self.getLatDiff(lat, bbox[0]) > tDiff){  // N
        tDiff = self.getLatDiff(lat, bbox[0]);
        edgeTweets[0] = tweet;
}
if(self.getLatDiff(lat, bbox[1]) > bDiff){ // S
        bDiff = self.getLatDiff(lat, bbox[1]);
        edgeTweets[1] = tweet;
}
if(self.getLonDiff(lon, bbox[2]) > rDiff){ // E
        rDiff = self.getLonDiff(lon, bbox[2]);
        edgeTweets[2] = tweet;
}
if(self.getLonDiff(lon, bbox[3]) > lDiff){ // W
        lDiff = self.getLonDiff(lon, bbox[3]);
        edgeTweets[3] = tweet;
}
if((self.getLatDiff(lat, bbox[0]) + self.getLonDiff(lon, bbox[3])) > tlDiff) {
// Top Left
        tlDiff = self.getLatDiff(lat, bbox[0]) + self.getLonDiff(lon, bbox[3]);
        edgeTweets[4] = tweet;
}
if((self.getLatDiff(lat, bbox[0]) + self.getLonDiff(lon, bbox[2])) > trDiff) {
// Top Right
        trDiff = self.getLatDiff(lat, bbox[0]) + self.getLonDiff(lon, bbox[2]);
        edgeTweets[5] = tweet;
}
if((self.getLatDiff(lat, bbox[1]) + self.getLonDiff(lon, bbox[3])) > blDiff) {
// Bottom Left
        blDiff = self.getLatDiff(lat, bbox[1]) + self.getLonDiff(lon, bbox[3]);
        edgeTweets[6] = tweet;
}
if((self.getLatDiff(lat, bbox[1]) + self.getLonDiff(lon, bbox[2])) > brDiff) {
// Bottom Right
        brDiff = self.getLatDiff(lat, bbox[1]) + self.getLonDiff(lon, bbox[2]);
        edgeTweets[7] = tweet;
}
```

ภาพที่ 3.19 การหาข้อความทวีตที่อยู่ขอบนอกสุดในพื้นที่

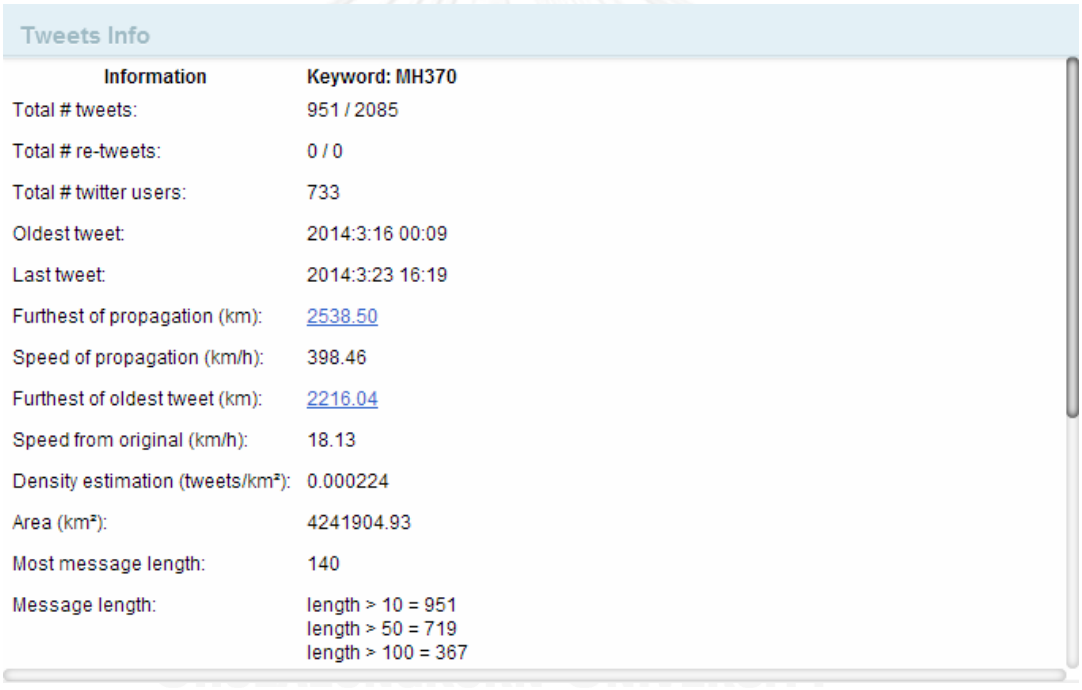
```
getDistance: function(tweetA, tweetB) {
    var R = 6371; // km
    var dLat = self.toRad(tweetB.lat - tweetA.lat);
    var dLon = self.toRad(tweetB.lon - tweetA.lon);
    var lat1 = self.toRad(tweetA.lat);
    var lat2 = self.toRad(tweetB.lat);

    var a = Math.sin(dLat/2) * Math.sin(dLat/2) +
            Math.sin(dLon/2) * Math.sin(dLon/2) * Math.cos(lat1) * Math.cos(lat2);
    var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
    return R * c;
},
```

ภาพที่ 3.20 การหาระยะทางด้วย Haversine formula

เมื่อได้ระยะทางออกมาแล้วจึงนำไปวัดเพื่อหาความเร็วในการแพร่กระจายโดยใช้หลักการหาความเร็วตามปกติคือระยะทางต่อหนึ่งหน่วยเวลา ทำให้ได้ความเร็วในการแพร่กระจายข่าวระหว่างทวีตที่ไกลที่สุดและความเร็วของการแพร่กระจายจากจุดเริ่มต้นว่าไกลออกไปเท่าใด

หาพื้นที่บนหน้าต่างกูเกิลเอิร์ธโดยใช้ BBOX มาคำนวณด้วย getDistance function แล้วใช้สูตรกว้างคูณยาวแบบการหาพื้นที่ของสี่เหลี่ยมผืนผ้า ซึ่งได้ทำการเปรียบเทียบกับฟังก์ชันการหาพื้นที่ของกูเกิลแล้วว่ามีความแตกต่างกันเพียงเล็กน้อยเท่านั้นขึ้นอยู่กับความสูงของมุมกล้องวัดจากระดับน้ำ หากมุมกล้องอยู่สูงหรือห่างออกไปมากๆ ความผิดพลาดก็จะมีเพิ่มขึ้นตามไปด้วย ผู้วิจัยจึงนำวิธีนี้มาใช้หาพื้นที่แล้วหาความหนาแน่นของทวีตในพื้นที่นั้นโดยคำนวณจากจำนวนทวีตต่อหนึ่งหน่วยตารางกิโลเมตร

สุดท้ายหาความยาวข้อความที่นิยมกันมากที่สุดโดยนับจำนวนข้อความที่เกิดขึ้นของแต่ละความยาวเก็บไว้ แล้วจึงนำมาเปรียบเทียบกัน ซึ่งเมื่อรวบรวมข้อมูลที่ต้องการครบถ้วนแล้วจะส่งค่าต่างๆ ขึ้นไปแสดงบนหน้าเพจในรูปแบบตารางดังรูปที่ 3.21


Tweets Info

Information	Keyword: MH370
Total # tweets:	951 / 2085
Total # re-tweets:	0 / 0
Total # twitter users:	733
Oldest tweet:	2014:3:16 00:09
Last tweet:	2014:3:23 16:19
Furthest of propagation (km):	2538.50
Speed of propagation (km/h):	398.46
Furthest of oldest tweet (km):	2216.04
Speed from original (km/h):	18.13
Density estimation (tweets/km²):	0.000224
Area (km²):	4241904.93
Most message length:	140
Message length:	length > 10 = 951 length > 50 = 719 length > 100 = 367

ภาพที่ 3.21 Tweets Info

3.3.5 การเก็บข้อมูลออฟไลน์

เมื่อไคลเอนต์ได้รับข้อความทวีตปริมาณหนึ่งในรูปแบบ JSON message จากฝั่งเซิร์ฟเวอร์ ไคลเอนต์จะต้องทำการจัดเก็บข้อความทวีตนั้นเพื่อนำเข้าประมวลไว้ในที่ๆหนึ่ง เพื่อให้ทุกมอดูลสามารถเข้าถึงข้อมูลได้ ทั้งนี้ผู้วิจัยได้ดึง Indexed DB หรือ Index Database API มาใช้เพื่อจัดเก็บข้อมูลลงไป

เริ่มต้นจะสร้าง database ขึ้นมาชื่อว่า “buffers” บ่งบอกว่าใช้เก็บข้อมูลเพียงชั่วคราวเท่านั้น จากนั้นจึงสร้างที่เก็บข้อมูล (Object storage) ในฐานข้อมูลหรือ database ชื่อว่า “statuses” ดังรูปที่ 3.22 โดยจะมีการตรวจสอบก่อนเสมอว่ามี Object storage “statuses” อยู่หรือไม่ หากมีอยู่จึงทำลายทิ้งแล้วสร้างใหม่อีกครั้ง

```
init: function() {
    var openRequest = indexedDB.open("buffers", 10);
    //indexedDB.deleteDatabase(dbName);
    openRequest.onerror = function(e) {
      console.log("Buffer error: " + e.target.errorCode);
      indexedDB.deleteDatabase(dbName);
    };
    openRequest.onsuccess = function(e) {
        dbObj = openRequest.result;
        console.log("Buffer: created");
    };
    openRequest.onupgradeneeded = function (e) {
        if(e.currentTarget.result.objectStoreNames.contains("statuses")) {
          e.currentTarget.result.deleteObjectStore("statuses");
        }
        var objectStore = e.currentTarget.result.createObjectStore("statuses", {keyPath: "id"});
        objectStore.createIndex("keywordIndex", "keyword", { unique: false });
        console.log("Buffer: upgraded");
    };
},
```

ภาพที่ 3.22 การสร้าง Indexed DB ฝั่งไคลเอนต์

ทำการเปิด Transaction เพื่อให้ระบบสามารถเข้าถึงฐานข้อมูลและจัดการกับข้อมูลได้ตามต้องการ ซึ่งในที่นี้จะเปิด transaction เป็น readwrite mode ให้อนุญาตเพิ่มข้อมูลเข้าที่เก็บและอ่านขึ้นมาเพื่อประมวลผล และเตรียมฟังก์ชั่นสำหรับยกเลิก transaction ดังกล่าวเมื่อจบงานค้นหาหนึ่งๆ ตามรูปที่ 3.23

```
openTransaction: function() {
    tx = dbObj.transaction("statuses", "readwrite");
    store = tx.objectStore("statuses");
},

abortTransaction: function() {
    store.abort();
},
```

ภาพที่ 3.23 การสร้างและทำลาย Transaction เพื่อเข้าถึง Object storage

หลังจากนั้นจะขึ้นอยู่กับการเรียกใช้งานว่ามอดูลต่างๆ ต้องการข้อมูลอะไรจากฐานข้อมูลนี้ โดยตัวจัดการฝั่งไคลเอนต์จะเป็นมอดูลที่ทำการ push ข้อมูลลงไปใน storage ส่วนมอดูลอื่นๆ จะทำการ pull ข้อมูลขึ้นมาใช้อีกที ดังฟังก์ชันที่จัดเตรียมให้ดังนี้

- Push function: นำข้อมูล JSON object ของแต่ละทวีตใส่ลงใน storage

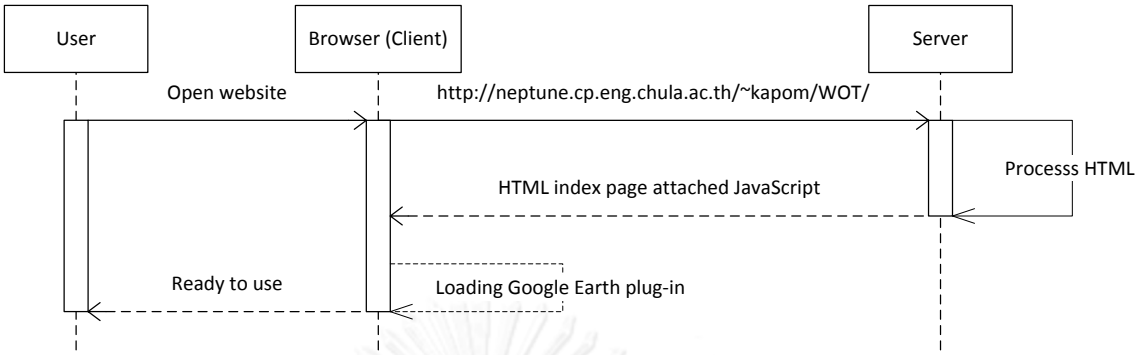
- pullEach function: สร้าง transaction ด้วย readonly mode เพื่อใช้สำหรับการดึงข้อมูลออกมาจาก storage โดยจะอาศัยการเลื่อน cursor แบบลงล่าง (descending) ในการดึงข้อมูลออกมาทีละ record ซึ่งเตรียมไว้ใช้กับกรณีเรียกดูข้อมูลแบบ all tweets
- pullEachByKeyword function: มีโครงสร้างหลักเหมือนกับ pullEach function แต่จะอาศัย IDBKeyRange ในการจำกัด index ที่ค้นหา ทำให้ cursor มองเห็นข้อมูลเท่าที่ต้องการเท่านั้น เพื่อใช้กับกรณีสืบค้นข้อมูลด้วยคีย์เวิร์ด
- countTweetsByKeywordAndTimeRange function: ใช้นับจำนวนทวีตของหัวข้อที่ค้นหาซึ่งอยู่ในช่วงเวลาที่กำหนด
- getOldestAndLastTweetByKeyword function: อาศัยหลักการเลื่อน cursor จัดเก็บทวีตที่เก่าที่สุดและใหม่ที่สุด โดย record แรกจะถือเป็นทวีตที่เก่าที่สุดส่วน record สุดท้ายจะถือเป็นทวีตที่ใหม่ที่สุด เนื่องจากข้อมูลถูกเรียงลำดับเก่าใหม่โดยเซิร์ฟเวอร์แต่เริ่มต้น

3.4 แผนภาพการทำงานของระบบ WOT

แผนภาพการทำงานนี้จะช่วยให้ผู้อ่านมองเห็นการทำงานของระบบได้ง่ายยิ่งขึ้น ซึ่งผู้พัฒนาได้รวบรวมฟังก์ชันการทำงานหลักๆ ของระบบที่ได้กล่าวไปแล้วเอาไว้ ดังนี้

3.4.1 การใช้งานเริ่มต้น

สำหรับการใช้งานเริ่มต้น ผู้ใช้ระบบจะต้องทำการเปิดหน้าเว็บไซต์ผ่านเบราว์เซอร์ขึ้นมาเพื่อร้องขอการใช้งานระบบ ซึ่งเซิร์ฟเวอร์เมื่อได้รับคำร้องขอจะจัดส่งหน้า index.php ที่ใช้ในการแสดงหน้าเพจ HTML กลับไปให้โดยภายในจะมี JavaScript ที่จำเป็นแนบไปด้วย เมื่อได้รับหน้าเพจแล้วเบราว์เซอร์จะจดจำสคริปต์นั้นเอาไว้และทำการดาวน์โหลดปลั๊กอินของกูเกิลเอิร์ธตามที่ HTML อ้างอิงถึง หลังจากนั้นผู้ใช้จึงสามารถใช้งานระบบได้ ตามรูปที่ 3.24



ภาพที่ 3.24 Sequence diagram การใช้งานเริ่มต้น

3.4.2 การค้นหาข้อความ

การค้นหานั้นมีด้วยกัน 2 แบบใหญ่ๆ คือการค้นหาแบบใช้คีย์เวิร์ดหรือไม่ใช้คีย์เวิร์ดหรือก็คือการค้นหาแบบ All tweets ซึ่งจะเก็บทวีตทั้งหมดในพื้นที่มาแสดงและวิเคราะห์บนเบราว์เซอร์ ทั้งนี้ทั้งสองแบบจะใช้หลักการเดียวกันในการทำงาน เพียงแต่เรียกหาฟังก์ชันการทำงานต่างกัน และมีการทำงานผสานหลายมอดูลในฝั่งไคลเอนต์และเซิร์ฟเวอร์ ดังจะเห็นในรูปที่ 3.25

มอดูลที่เกี่ยวข้องทางฝั่งไคลเอนต์ ได้แก่

- Client operation: เพื่อจัดรูปแบบ URL ในการร้องขอไปให้เซิร์ฟเวอร์และประสานงานกับมอดูลต่างๆเพื่อวิเคราะห์ข้อมูลที่ได้รับจากเซิร์ฟเวอร์
- Google Earth module (GE): เพื่อจำลองข้อมูลทวีตบนหน้าเพจในรูปแบบของแผนที่
- Google Chart module (GC): ประมวลชุดข้อมูลเพื่อหาแนวโน้มในรูปแบบแผนภูมิเส้น
- Tweet analytics: วิเคราะห์ชุดข้อมูลเชิงพื้นที่เพื่อนำเสนอข้อมูลในลักษณะตัวเลข
- Buffer: จัดเก็บชุดข้อมูลที่ได้รับจากเซิร์ฟเวอร์ลงใน Indexed DB

มอดูลที่เกี่ยวข้องทางฝั่งเซิร์ฟเวอร์ ได้แก่

- AJAX module: เพื่อบอกกับตัวจัดการกลางว่าคำร้องขอจากไคลเอนต์เกี่ยวกับเรื่องใด
- Operation manager: ตัวจัดการฝั่งเซิร์ฟเวอร์ที่ติดต่อกับ database manager เพื่อจัดหาข้อมูลที่เหมาะสมของแต่ละคำร้อง ซึ่งการค้นแบบใช้คีย์เวิร์ดและเลือกเอาทั้งหมดนั้นจะไปเรียกฟังก์ชันของ database manager ต่างกัน

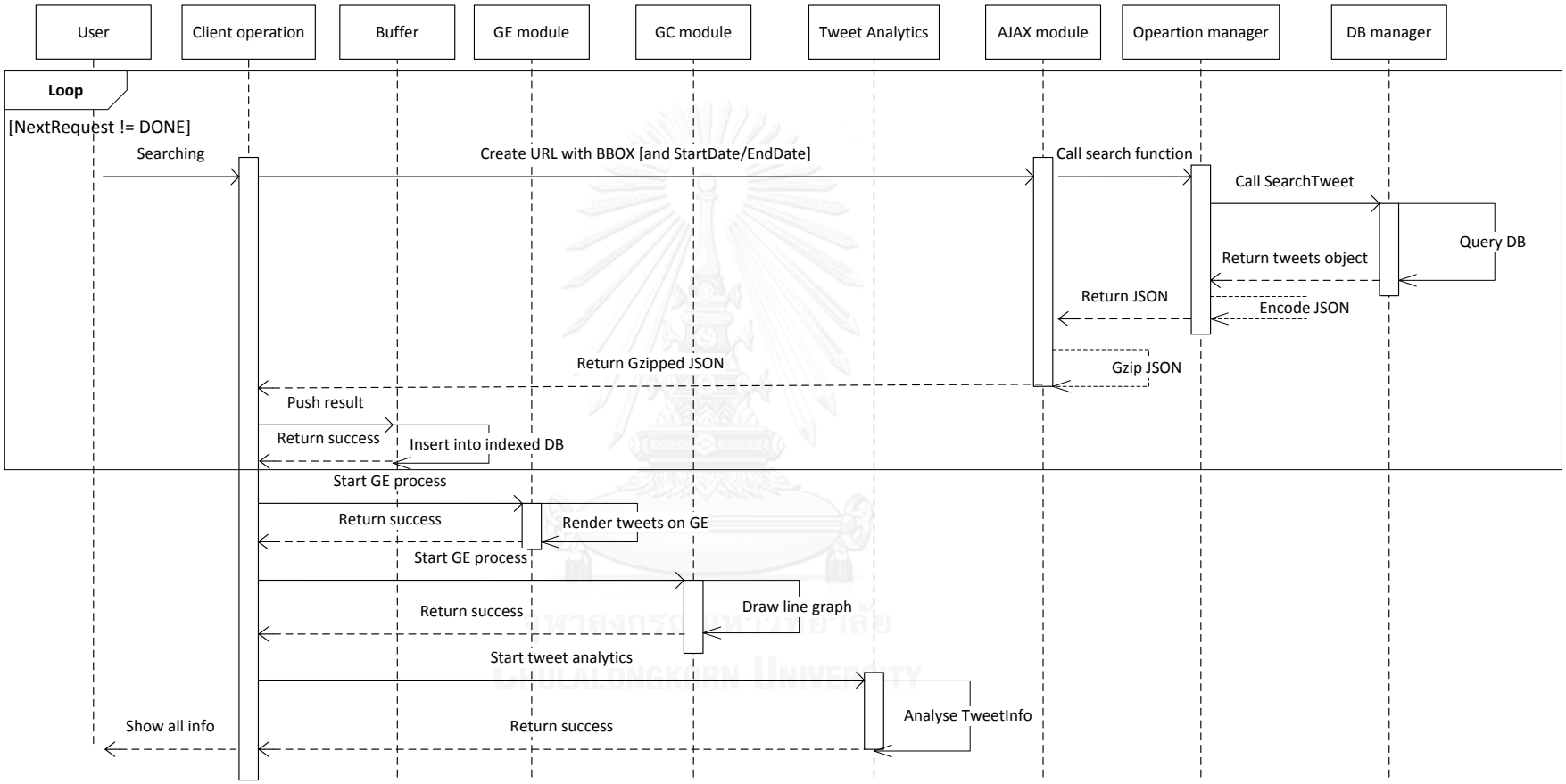
- Database manager (DB): ติดต่อฐานข้อมูล “testbed” และสร้าง SQL statement เพื่อไปเรียกค้นฐานข้อมูล

3.4.3 การจัดเก็บข้อมูลถาวร (Archive mode)

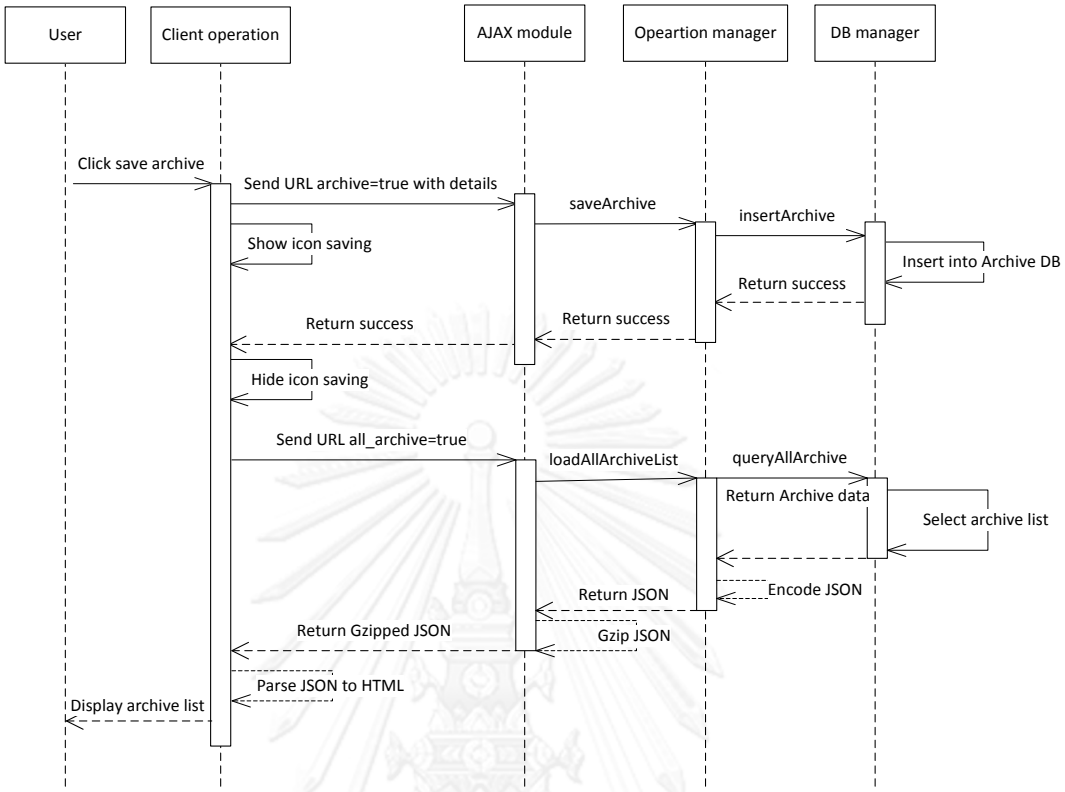
การจัดเก็บข้อมูลถาวรนี้เป็นโหมดที่ช่วยให้ผู้ใช้สามารถจัดเก็บผลลัพธ์ของการค้นหาครั้งนั้นๆ เอาไว้ดูในภายหลังได้โดยไม่ถูก House cleaner ของระบบลบข้อมูลออกไป ในส่วนนี้จะมี 3 activity ที่ใช้ทำงาน คือ การจัดเก็บลงฐานข้อมูลถาวร (Save archive) การแสดงข้อมูลที่ถูกจัดเก็บ (Load archive) และลบข้อมูลจากฐานข้อมูลถาวร (Delete archive) ทั้งนี้ส่วนของการแสดงข้อมูลจะมีมอดูลที่เกี่ยวข้องและการทำงานเหมือนกันกับหัวข้อย่อย 3.4.2 เพียงแต่การเรียกใช้ Database manager จะเรียกไปที่ฟังก์ชันเฉพาะสำหรับ Archive mode

สำหรับการจัดเก็บและการลบข้อมูลออกจากฐานข้อมูลถาวรนั้นจะมีกระบวนการการทำงานเป็นไปตามรูปที่ 3.26 และ 3.27 โดยมีส่วนที่เกี่ยวข้องดังนี้

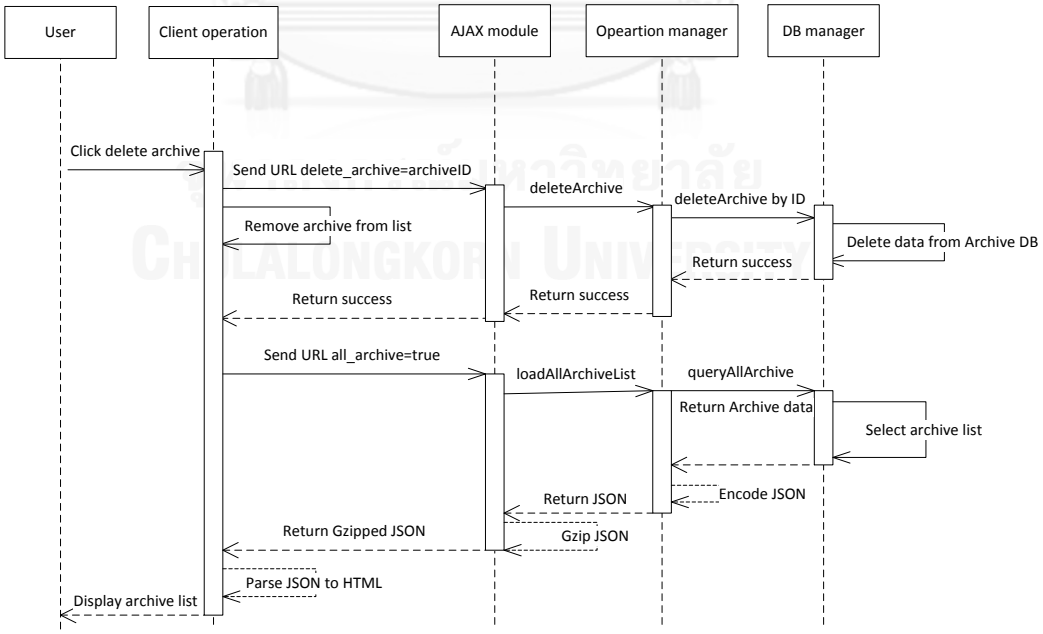
- Client operation: สร้าง URL สำหรับเรียกใช้งานฟังก์ชันการจัดเก็บและลบข้อมูล
- AJAX module: เพื่อบอกกับตัวจัดการกลางว่าคำร้องขอจากไคลเอนต์เกี่ยวกับเรื่องใด
- Operation manager: ติดต่อกับ database manager เพื่อเรียกฟังก์ชันการจัดเก็บหรือลบข้อมูลจาก archive
- Database manager (DB): ติดต่อฐานข้อมูล “testbed” และสร้าง SQL statement เพื่อไปจัดเก็บหรือลบข้อมูลบนฐานข้อมูล Archive



ภาพที่ 3.25 Sequence diagram ของการค้นหา (Searching)



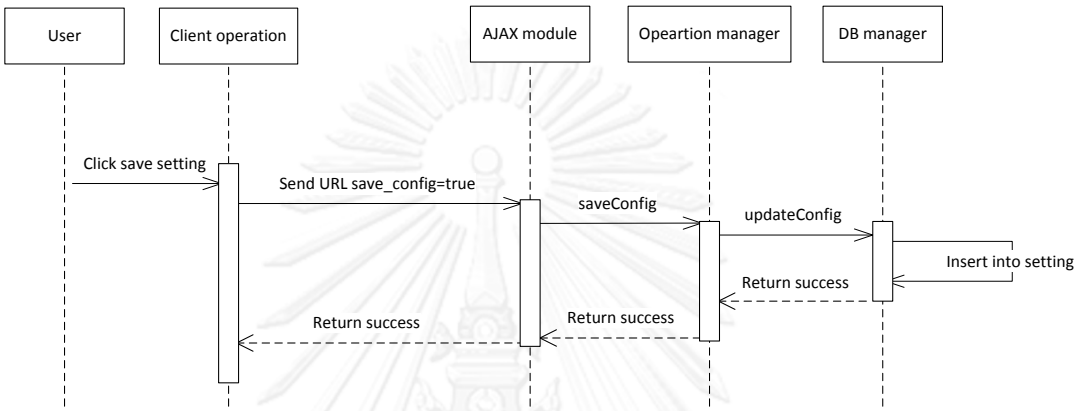
ภาพที่ 3.26 Sequence diagram ของการบันทึกข้อมูลลงฐานข้อมูลถาวร



ภาพที่ 3.27 Sequence diagram ของการลบ archive ออกจาก list

3.4.4 การตั้งค่า

การตั้งค่า Tweet download และ Number of keywords ผ่านหน้าเว็บเพจนั้น จะต้องทำการส่งไปเก็บบนฐานข้อมูลฝั่งเซิร์ฟเวอร์ทุกครั้ง เพราะส่วนที่จะเรียกไปใช้งานคือ Operation manager ซึ่งเป็นมอดูลหนึ่งบนเซิร์ฟเวอร์ ทั้งนี้จะมีการทำงานดังรูปที่ 3.28



ภาพที่ 3.28 Sequence diagram ของการปรับแต่งค่าการตั้งค่าบนระบบ

บทที่4
การทดลอง

ในบทนี้จะนำเสนอการทดลองใช้งานระบบ WOT ตั้งแต่ส่วนการเรียกคืนทวีตจากทวิตเตอร์และการค้นหา รวมถึงการวิเคราะห์และจัดแสดงผลข้อมูลในเชิงพื้นที่และเวลาต่างๆ โดยจะแสดงให้เห็นพฤติกรรมบางอย่างแตกต่างกันด้วยคำค้นหาที่อิงหัวข้อสนทนาที่แตกต่างกัน และทำการทดสอบแต่ละมอดูลเพื่อยืนยันความน่าเชื่อถือของข้อมูล

4.1 การทดลอง

การทดลองจะทำการจำกัดขอบเขตให้อยู่ในพื้นที่ประเทศไทยซึ่งครอบคลุมบริเวณระหว่างละติจูด 5.610000 ถึง 20.463181 และลองจิจูดระหว่าง 97.345619 ถึง 105.6393811 และเก็บข้อมูลไว้ในฐานข้อมูล 7 วันโดยมีสคริปต์ลบข้อความที่เก่ากว่า 7 วันทุกเที่ยงคืนเนื่องด้วยข้อจำกัดของพื้นที่บนเซิร์ฟเวอร์ ทั้งนี้สามารถปรับเปลี่ยนขอบเขตได้ตามต้องการโดยเข้าไปปรับแต่งค่าละติจูดและลองจิจูดในไพธอนสคริปต์บนเครื่องเซิร์ฟเวอร์

การทดลองจะดำเนินการโดยผู้ทดสอบซึ่งจะประพฤติตัวเสมือนผู้ใช้งานทั่วไปที่เข้าใช้งานระบบและเรียกค้นหัวข้อที่สนใจและดูผลผ่านเบราว์เซอร์ โดยจะเห็นว่าข้อความที่เกี่ยวข้องกับหัวข้อนั้นๆถูกจัดแสดงเป็นหมุดบนแผนที่กูเกิลเอิร์ธแสดงการกระจายตัวของข้อมูลในพื้นที่ และแสดงผลการวิเคราะห์เชิงสถิติและพื้นที่ ซึ่งจะอธิบายต่อไป

4.1.1 ปริมาณข้อมูลทวีต

ในช่วงเริ่มต้นได้ทดลองดึงข้อความทวีตบนโลกทั้งหมด คือ ช่วงละติจูด -90.000000 ถึง 90.000000 และลองจิจูด -180.000000 ถึง 180.000000 ตลอด 7 วันทำให้ได้รับข้อความประมาณ 13 ล้านข้อความมาเก็บไว้ในระบบ ซึ่งกินพื้นที่กว่า 6.3 GB หรือ 60% ของพื้นที่ในdirectory /var ซึ่งเป็นพื้นที่ที่เซิร์ฟเวอร์ได้รับการติดตั้ง MySQL สำหรับเก็บข้อมูลเอาไว้ ส่งผลให้การค้นหาข้อมูลจากฐานข้อมูลใช้เวลานาน ดังนั้นจึงลดขอบเขตการเก็บข้อความลงให้ครอบคลุมพื้นที่ในเขตพื้นที่ประเทศไทยเพื่อใช้ศึกษาแทน ซึ่งจะได้จำนวนทวีต ประมาณ 1 ล้านข้อความต่อหนึ่งอาทิตย์ หรือคิดเป็น 0.14 ล้านข้อความต่อวัน ทั้งนี้ข้อความดังกล่าวได้ถูกคัดกรองข้อความที่ไม่พึงประสงค์ออกโดยจะคัดเฉพาะข้อความที่มีพิกัดที่อยู่ ข้อความที่มีขนาดไม่สั้นจนเกินไป (ในที่นี้จำกัดที่ความยาวมากกว่า 14

ตัวอักษร หรือมากกว่า 10% ของความยาวสูงสุด 140 ตัวอักษร) และคัดข้อความที่ประกอบด้วยคำไม่สุภาพหรือคำหยาบซึ่งได้มาจาก www.wdyl.com ออก ชุดคำหยาบที่ระบบใช้มีตามรูปที่ 4.1

```
swearwords = ["4r5e", "5h1t", "5hit", "a55", "anal", "anus", "ar5e", "arrse", "arse", "ass", "ass-fucker", "asses",
"assfucker", "assfukka", "asshole", "assholes", "asswhole", "a_s_s", "b!tch", "b00bs", "b17ch", "b1tch", "ballbag",
"balls", "ballsack", "bastard", "beastial", "beastiality", "bellend", "bestial", "bestiality", "bi+ch", "biatch",
"bitch", "bitcher", "bitchers", "bitches", "bitchin", "bitching", "bloody", "blow job", "blowjob", "blowjobs", "boiolas",
"bollock", "bollok", "boner", "boob", "boobs", "booobs", "boooobs", "booooobs", "boooooobs", "breasts", "buceta",
"bugger", "bum", "bunny fucker", "butt", "butthole", "buttmuch", "buttplug", "c0ck", "c0cksucker", "carpet muncher",
"cawk", "chink", "cipa", "cl1t", "clit", "clitoris", "clits", "cnut", "cock", "cock-sucker", "cockface", "cockhead",
"cockmunch", "cockmuncher", "cocks", "cocksuck", "cocksucked", "cocksucker", "cocksucking", "cocksucks", "cocksuka",
"cocksukka", "cok", "cokmuncher", "coksucka", "coon", "cox", "crap", "cum", "cummer", "cumming", "cums", "cumshot",
"cunilingus", "cunillingus", "cunnilingus", "cunt", "cuntlick", "cuntlicker", "cuntlicking", "cunts", "cyalis",
"cyberfuc", "cyberfuck", "cyberfucked", "cyberfucker", "cyberfuckers", "cyberfucking", "d1ck", "damn", "dick",
"dickhead", "dildo", "dildos", "dink", "dinks", "dirsa", "dlck", "dog-fucker", "doggin", "dogging", "donkeyribber",
"doosh", "duche", "dyke", "ejaculate", "ejaculated", "ejaculates", "ejaculating", "ejaculatings", "ejaculation",
"ejakulate", "f u c k", "f u c k e r", "f4nny", "fag", "fagging", "faggitt", "faggot", "faggs", "fagot", "fagots",
"fags", "fanny", "fannyflaps", "fannyfucker", "fanyy", "fatass", "fcuk", "fcuker", "fcuking", "feck", "fecker",
"felching", "fellate", "fellatio", "fingerfuck", "fingerfucked", "fingerfucker", "fingerfuckers", "fingerfucking",
"fingerfucks", "fistfuck", "fistfucked", "fistfucker", "fistfuckers", "fistfucking", "fistfuckings", "fistfucks",
"flange", "fook", "fooker", "fuck", "fucka", "fucked", "fucker", "fuckers", "fuckhead", "fuckheads", "fuckin", "fucking",
"fuckings", "fuckingshitmotherfucker", "fuckme", "fucks", "fuckwhit", "fuckwit", "fudge packer", "fudgepacker", "fuk",
"fuker", "fukker", "fukkin", "fuks", "fukwhit", "fukwit", "fux", "fux0r", "f_u_c_k", "gangbang", "gangbanged",
"gangbangs", "gaylord", "gaysex", "goatse", "God", "god-dam", "god-damned", "goddamn", "goddamned", "hardcoresex",
"hell", "heshe", "hoar", "hoare", "hoer", "homo", "hore", "horniest", "horny", "hotsex", "jack-off", "jackoff", "jap",
"jerk-off", "jism", "jiz", "jizm", "jizz", "kawk", "knob", "knobead", "knobed", "knobend", "knobhead", "knobjocky",
"knobjokey", "kock", "kondum", "kondums", "kum", "kummer", "kumming", "kums", "kunilingus", "l3i+ch", "l3itch", "labia",
"lmfao", "lust", "lusting", "m0f0", "m0fo", "m45terbate", "ma5terb8", "ma5terbate", "masochist", "master-bate",
"masterb8", "masterbat*", "masterbat3", "masterbate", "masterbation", "masterbations", "masturbate", "mo-fo", "mof0",
"mofo", "mothafuck", "mothafucka", "mothafuckas", "mothafuckaz", "mothafucked", "mothafucker", "mothafuckers",
"mothafuckin", "mothafucking", "mothafuckings", "mothafucks", "mother fucker", "motherfuck", "motherfucked",
"motherfucker", "motherfuckers", "motherfuckin", "motherfucking", "motherfuckings", "motherfuckka", "motherfucks",
"muff", "mutha", "muthafecker", "muthafuckker", "muther", "mutherfucker", "n1gga", "n1gger", "nazi", "nigg3r", "nigg4h",
"nigga", "niggah", "niggas", "niggaz", "nigger", "niggers", "nob", "nob jokey", "nobhead", "nobjocky", "nobjokey",
"numbnuts", "nutsack", "orgasim", "orgasims", "orgasm", "orgasms", "p0rn", "pawn", "pecker", "penis", "penisfucker",
"phonesex", "phuck", "phuk", "phuked", "phuking", "phukked", "phukking", "phuks", "phuq", "pigfucker", "pimpis", "piss",
"pissed", "pisser", "pissers", "pisses", "pissflaps", "pissin", "pissing", "pissoff", "poop", "porn", "porno",
"pornography", "pornos", "prick", "pricks", "pron", "pube", "pusse", "pussi", "pussies", "pussy", "pussys", "rectum",
"retard", "rimjaw", "rimming", "s hit", "s.o.b.", "sadist", "schlong", "screwing", "scroat", "scrote", "scrotum",
"semen", "sex", "sh!+", "sh!t", "sh1t", "shag", "shagger", "shaggin", "shagging", "shemale", "shit", "shitdick", "shite",
"shited", "shitey", "shitfuck", "shitfull", "shithead", "shiting", "shitings", "shits", "shitted", "shitter", "shitters",
"shitting", "shittings", "shitty", "skank", "slut", "sluts", "smegma", "smut", "snatch", "son-of-a-bitch", "spac",
"spunk", "s_h_i_t", "t1tt1e5", "t1tties", "teets", "teez", "testical", "testicle", "tit", "titfuck", "tits", "titt",
"tittie5", "tittiefucker", "titties", "tittyfuck", "tittywank", "titwank", "tosser", "turd", "tw4t", "twat", "twathead",
"twatty", "twunt", "twunter", "v14gra", "v1gra", "vagina", "viagra", "vulva", "w00se", "wang", "wank", "wanker", "wanky",
"whoar", "whore", "willies", "willy", "xrated", "xxx"]
```

ภาพที่ 4.1 ชุดคำหยาบที่ใช้ในการคัดกรองทวีต

4.1.2 การค้นหาข้อความ

การค้นหาข้อความหรือหัวข้อที่สนใจจากระบบนั้น สามารถทำได้ทั้งหมดสองแบบด้วยกันคือ ค้นหาด้วยคีย์เวิร์ดที่สนใจ หรือเลือกค้นหาทวีตทั้งหมดที่เกิดขึ้นในพื้นที่ ซึ่ง URL ที่ Client operation ส่งให้จะประกอบด้วยชุดพารามิเตอร์ที่ต่างกันเพื่อบ่งชี้ว่าผู้ใช้งานต้องการข้อมูลแบบใด

URL สำหรับการค้นหาด้วยคีย์เวิร์ดจะประกอบด้วย

- Keyword=<words> : กรณีที่ไม่กรอกจะถือเป็นการร้องขอทวีตทั้งหมด
- bbox=<LAT_{TR}, LAT_{BL}, LON_{TR}, LON_{BL}> : พื้นที่ตาม View ในกูเกิลเอิร์ธปลั๊กอิน
- startdate=<datetime> : Optional parameter
- enddate=<datetime> : Optional parameter
- searchmode=<user or message> : message เป็นค่า default

ตัวอย่างการค้นหาคำว่า “test” ในฐานะข้อความ (message) บนพื้นที่ BBOX = [111.30675804430933, 21.09746162609411, 89.97776711537985, 4.468816709619943] ไม่จำกัดช่วงเวลา จะทำให้ได้ URL ดังนี้

ajax.php?keyword=test&bbox=21.09746162609411,4.468816709619943,111.30675804430933,89.97776711537985&startdate=&enddate=&searchmode=msg

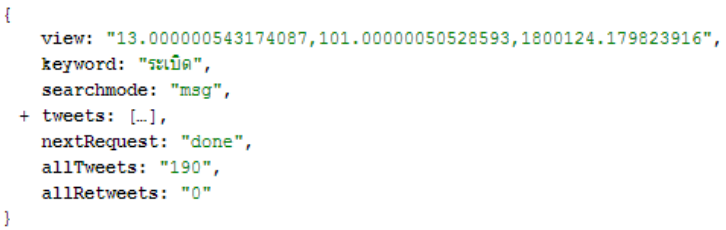
URL สำหรับการค้นหาทวีตทั้งหมดบนพื้นที่จะประกอบด้วย

- all_tweet=true : บ่งบอกว่าเป็นการค้นหาทวีตทั้งหมด
- bbox=<LAT_{TR}, LAT_{BL}, LON_{TR}, LON_{BL}> : พื้นที่ตาม View ในกูเกิลเอิร์ธปลั๊กอิน
- startdate=<datetime> : Optional parameter
- enddate=<datetime> : Optional parameter

ตัวอย่างการค้นหาข้อความทั้งหมดบนพื้นที่ BBOX = [111.30675804430933, 21.09746162609411, 89.97776711537985, 4.468816709619943] ไม่จำกัดช่วงเวลา จะได้ URL ดังนี้

ajax.php?all_tweet=true&bbox=21.09746162609411,4.468816709619943,111.30675804430933,89.97776711537985&startdate=&enddate=

URL ดังกล่าวจะถูกส่งไปให้เซิร์ฟเวอร์ โดยมอดูลแรกของฝั่งเซิร์ฟเวอร์ที่ถูกเรียกคือ ajax.php เพื่อบ่งชี้ว่า URL ดังกล่าวเป็นการค้นหาประเภทใด แล้วจึงให้ตัวจัดการ operation manager ประสานงานกับส่วน database module เพื่อจัดสรรข้อมูลที่เหมาะสมและตอบกลับมาในรูป JSON message ดังรูปที่ 4.2 และ 4.3



ภาพที่ 4.2 JSON message ผลลัพธ์จากการค้นหาของเซิร์ฟเวอร์

```
tweets: [
 - {
       id: "450885860511068163-ระเบิด",
       text: "อึมจนหุงจะระเบิดแต่ก็สติลแดกต่อไป เหอะๆ",
       source: "Twitter for iPad",
       createAt: 1396309340,
       lat: "5.7724524",
       lon: "101.07129261",
       userId: "162791397",
       screenName: "Be2daOw",
       profileImgURL: "http://pbs.twimg.com/profile_images/3039813232/f33f150bb3f738e8ed2bde80ec6a06d3_normal.jpeg",
       retweet: "0",
       retweetCount: "0",
       keyword: "ระเบิด"
   },
 - {
       id: "450940935556173824-ระเบิด",
       text: "ปวดหัวอ่ะ ปวดมาก จะระเบิดตายล่ะ",
       source: "Twitter for iPhone",
       createAt: 1396322471,
       lat: "6.78957402",
       lon: "100.68998432",
       userId: "1422571687",
       screenName: "22_ponny",
       profileImgURL: "http://pbs.twimg.com/profile_images/450939824317276161/QT0abAaK_normal.jpeg",
       retweet: "0",
       retweetCount: "0",
       keyword: "ระเบิด"
   }
],
```

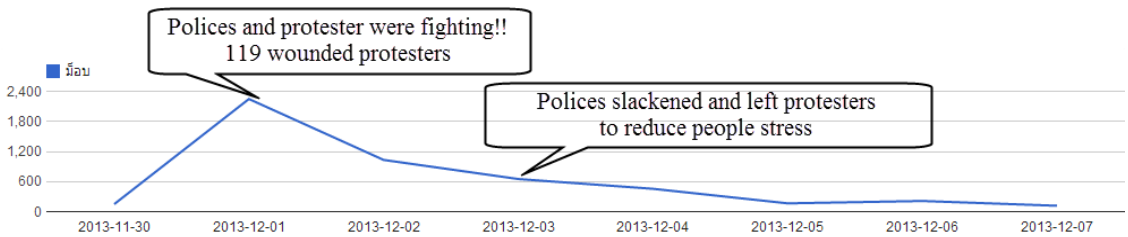

ภาพที่ 4.3 Array object ของข้อความทวีตใน JSON

4.1.3 การวิเคราะห์เชิงเวลา

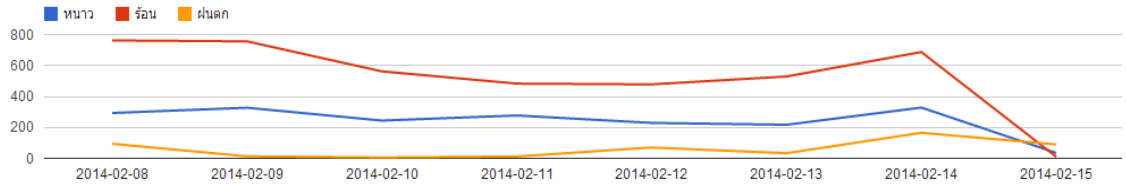
ในฝั่งไคลเอนต์ที่ได้รับ JavaScript จากเซิร์ฟเวอร์เพื่อใช้ประมวลผลชุดข้อมูลก่อนแสดงบนเบราว์เซอร์จะแบ่งออกเป็นการวิเคราะห์เชิงพื้นที่และเชิงเวลา ในส่วนนี้จะกล่าวถึงการวิเคราะห์ชุดข้อมูลโดยเปรียบเทียบกับหน่วยเวลาแสดงเป็นแผนภูมิเส้นซึ่งแสดงให้เห็นแนวโน้มของความนิยมได้ง่ายมากขึ้น

สำหรับส่วนของแกนเวลาจะเห็นผลลัพธ์ 2 แบบด้วยกัน คือ ต่อวัน และ ต่อชั่วโมง โดยเงื่อนไขในการเลือกแกนจะขึ้นกับ startdate และ enddate parameter ณ ขณะที่ผู้ใช้ค้นหาทวีต หากทำการค้นหาทวีตโดยระบุวันและเวลาใน startdate หรือ/และ enddate แผนภูมิที่วาดออกมาจะเป็นการเปรียบเทียบจำนวนข้อความทวีตในหนึ่งชั่วโมงตลอดช่วงเวลาที่ทำการขอเรียกค้น หากค้นหาทวีตโดยไม่ระบุวันเวลาไม่ว่าจะค้นหาด้วยคีย์เวิร์ดหรือไม่ แผนภูมิที่วาดออกมาจะเปรียบเทียบระหว่างจำนวนข้อความต่อหนึ่งวัน ซึ่งฐานข้อมูลจะเก็บข้อมูลเพียง 7 วันดังนั้นจึงทำให้เห็นผลลัพธ์เพียง 7 จุดข้อมูลเท่านั้น

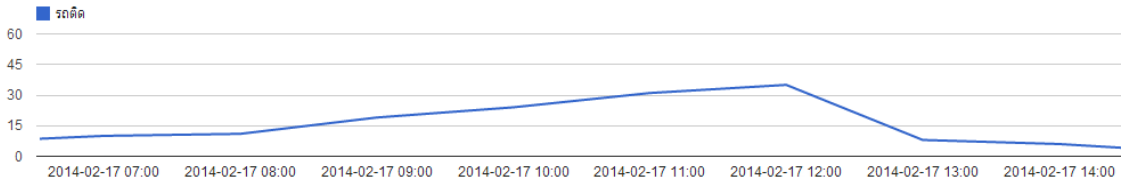
ในการทดลองผู้วิจัยทำการค้นหาด้วยคีย์เวิร์ดคำว่า “ม็อบ” “หนาว,ร้อน,ฝนตก” และ “รถติด” ซึ่งจะแสดงผลลัพธ์ที่แตกต่างกันออกไป โดยคีย์เวิร์ดแรกนั้นจะเกี่ยวกับเหตุการณ์ชุมนุมเมื่อปลายเดือนพฤศจิกายน ค.ศ. 2013 ในประเทศไทยที่ผ่านมา ซึ่งเหล่าผู้ชุมนุมออกมาต่อต้านรัฐบาลกันเป็นระยะเวลายาวนาน จนรัฐบาลเริ่มมีการเคลื่อนไหวเพื่อให้ประชาชนสลายการชุมนุม สังเกตได้จากรูปที่ 4.4 ว่าวันที่ 30 พฤศจิกายนมีจำนวนผู้ใช้ทวิตเตอร์พูดถึง “ม็อบ” เพียง 149 ข้อความ แต่ในวันที่ 1 ธันวาคมที่เกิดการปะทะกันระหว่างตำรวจและผู้ชุมนุมอย่างรุนแรงจนเกิดผู้บาดเจ็บกว่า 119 คนนั้นจำนวนทวีตก็พุ่งขึ้นสูงถึง 2,246 ข้อความ ในวันถัดมาเมื่อตำรวจยอมล่าถอยเพื่อคลายความตึงเครียดของผู้ชุมนุมอัตราการทวีตก็เริ่มลดลงตามลำดับเช่นกัน สำหรับคีย์เวิร์ดถัดมา “หนาว,ร้อน,ฝนตก” จะช่วยให้เห็นวิธีการเปรียบเทียบข้อมูลของระบบเมื่อผู้ใช้งานทำการค้นหาด้วยคีย์เวิร์ดมากกว่า 1 คำ จากรูปที่ 4.5 จะเห็นว่าแต่ละคีย์เวิร์ดถูกแทนด้วยเส้นแผนภูมิสีแตกต่างกัน เพื่อให้เปรียบเทียบได้ง่ายขึ้น ซึ่งจะเห็นอย่างชัดเจนว่ามีผู้ใช้ทวิตเตอร์กล่าวถึงคำว่า “ร้อน” มากกว่า “หนาว” และ “ฝนตก” มากตลอดอาทิตย์ และสุดท้ายคีย์เวิร์ด “รถติด” โดยทำการค้นหาร่วมกับช่วงวันเวลาที่สนใจ ทำให้เห็นความถี่ที่เกิดขึ้นในช่วงเวลาดังกล่าวว่า ช่วงเวลาระหว่าง 9 โมง เช้าถึงเที่ยงมีผู้ใช้ทวิตเตอร์พูดถึงเรื่องรถติดมากกว่าช่วงเวลาอื่นๆ ดังรูปที่ 4.6



ภาพที่ 4.4 แผนภูมิผลการค้นหาด้วยคีย์เวิร์ด “ม็อบ”



ภาพที่ 4.5 แผนภูมิผลการค้นหาด้วยคีย์เวิร์ด “หนาว,ร้อน,ฝนตก”

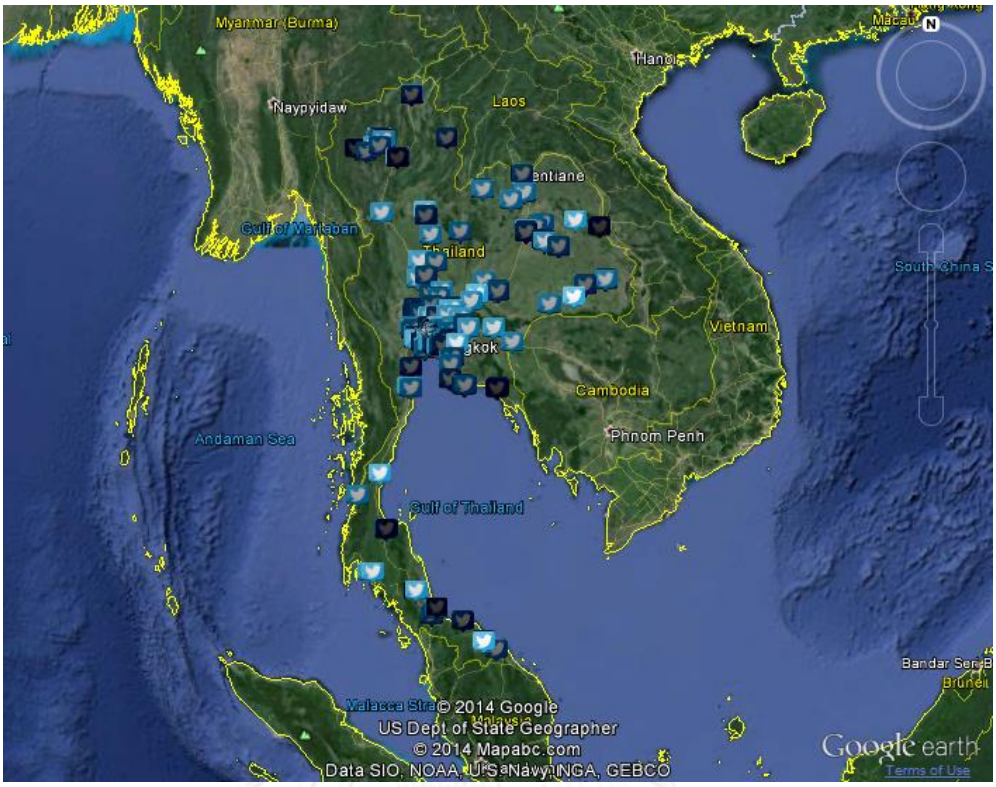


ภาพที่ 4.6 แผนภูมิผลการค้นหาด้วยคีย์เวิร์ด “รถติด”

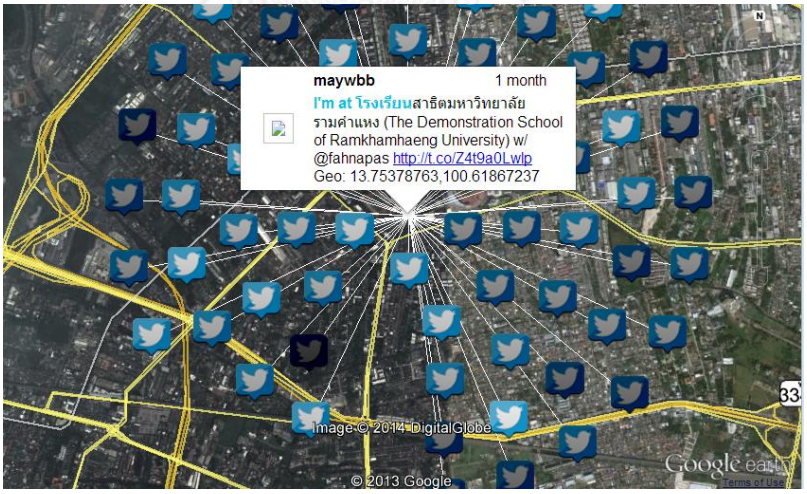
4.1.4 การวิเคราะห์เชิงพื้นที่

การแสดงตำแหน่งของข้อความบนแผนที่ช่วยให้เห็นการกระจายของผู้ใช้ทวิตเตอร์ในเชิงพื้นที่และการใช้ระดับความสว่างกับหมุดจะช่วยผนวกการแสดงข้อมูลเชิงเวลาเข้าด้วยกัน โดยความสว่างต่ำจะหมายถึงทวีตที่มีอายุมากส่วนความสว่างสูงจะหมายถึงทวีตที่ใหม่กว่า ทั้งนี้การปักหมุดข้อความจะทำให้เห็นถึงคุณลักษณะที่แตกต่างกันไปในแต่ละหัวข้อ เช่น การค้นหาด้วยคำว่า “I’m at โรงเรียน” ซึ่งผู้ใช้ทวิตเตอร์ทำการ check-in ผ่านแอพพลิเคชั่น Foursquare แล้วแชร์ข้อมูลไปบนทวิตเตอร์ด้วย ตามรูปที่ 4.7 จะเห็นความหนาแน่นของทวีตบริเวณกรุงเทพฯและปริมณฑลที่มีจำนวนโรงเรียนมากกว่าพื้นที่อื่นๆ และเมื่อขยายภาพเข้าไปจะเห็นการกระจุกตัวของทวีตจำนวนมหาศาล ณ พื้นที่หนึ่งๆ ซึ่งเกิดจากผู้ใช้โพสทวีตผ่านแอพพลิเคชั่นแล้วใช้ชื่อสถานที่ในการระบุพิกัด หรือที่เรียกกันว่า “Check-in” ดังรูปที่ 4.8

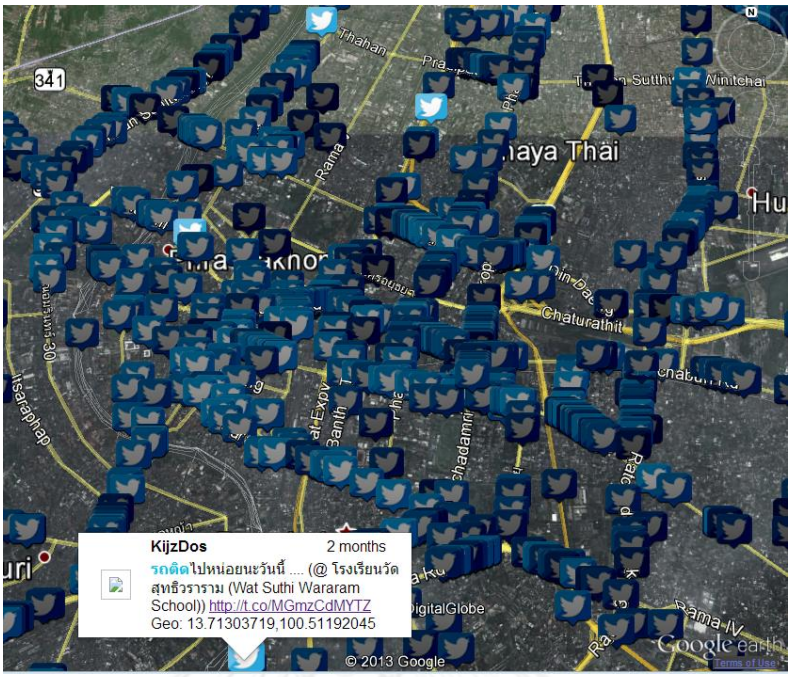
เมื่อเทียบกับการค้นหาด้วยคำว่า “รถติด” ที่แม้จะแสดงให้เห็นความหนาแน่นสูงของทวีตบริเวณกรุงเทพฯและปริมณฑลแบบเดียวกันก็ตาม แต่เมื่อขยายภาพเข้าไปจะเห็นว่าทวีตมีการกระจายตัวกันอยู่อย่างเป็นอิสระ ทั้งยังเรียงตัวตามแนวท้องถนนเนื่องจากผู้ใช้ทวิตเตอร์โพสข้อความด้วยตนเองและแนบพิกัดผ่านระบบ GPS โดยตรง ตามรูปที่ 4.9ส่วนการค้นหาด้วยหลายคีย์เวิร์ดจะแสดงให้เห็นการกระจายของทวีตด้วยสีที่แตกต่างกันเหมือนกับแผนภูมิเส้นซึ่งแสดงผลการวิเคราะห์เชิงเวลาตามรูปที่ 4.10



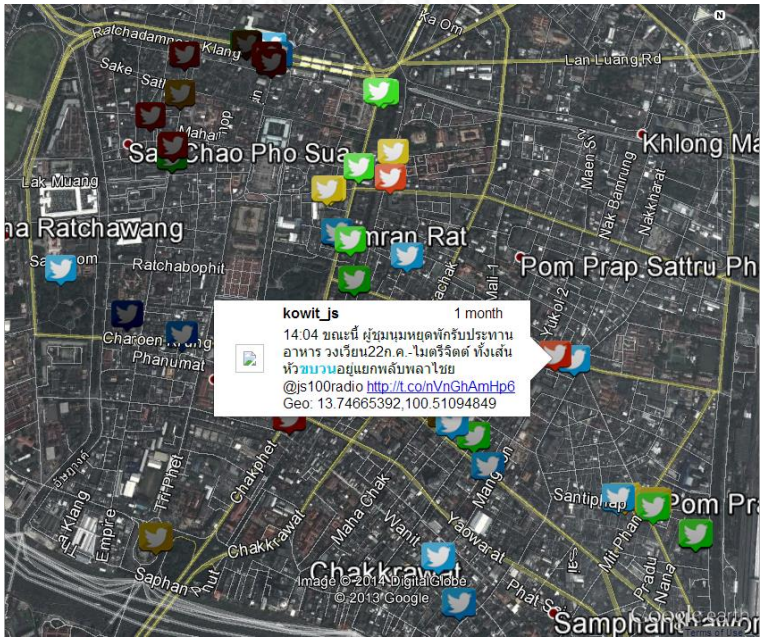
ภาพที่ 4.7 ความหนาแน่นทวีตของคำค้น “I’m at โรงเรียน”



ภาพที่ 4.8 ภาพขยายของคำค้น “I’m at โรงเรียน”



ภาพที่ 4.9 ภาพขยายของคำค้น “รถติด”



ภาพที่ 4.10 กูเกิลเอิร์ธแสดงผลการค้นหาด้วยหลายคีย์เวิร์ด

นอกจากนี้ระบบยังนำเสนอการคำนวณสถิติเชิงพื้นที่ที่ให้ผลลัพธ์เป็นตัวเลขช่วยให้ผู้ใช้งานเห็นข้อมูลที่ชัดเจนมากขึ้น เช่น ความเร็วในการแพร่กระจายของข่าวสาร ระยะทางระหว่างทวีตที่ไกลที่สุด หรือทวีตที่เก่าที่สุดกับใหม่ที่สุด ดังตัวอย่างในตารางที่ 4.1 ซึ่งเป็นผลลัพธ์จากการค้นหาคำว่า

“รถติด” ในกรอบพื้นที่ประเทศไทย ซึ่งครอบคลุมกว่า 3,724,389.62 ตร.กม. ในช่วง 2 วันคือ 17-19 กุมภาพันธ์ ค.ศ. 2014 ผู้ใช้งาน 350 คนทวีตข้อความที่เกี่ยวข้องถึง 913 ทวีตจากทั้งหมด 2,072,279 ทวีต โดยทวีตมีขอบเขตกว้าง 1,453.19 กม. และความเร็วในการแพร่กระจายไปยังจุดที่ไกลที่สุดคือ 37.7 กม./ชม. หากนับจากทวีตที่เก่าที่สุดไปใหม่ที่สุดจะเห็นว่าความเร็วในการเดินทางคือ 11.64 กม./ชม.

ตารางที่ 4.1 ผลการคำนวณสถิติเชิงพื้นที่ของคำค้น “รถติด”

Information	**Keyword: รถติด**
Total # tweets:	913 / 2072279
Total # twitter users:	350
Oldest tweet:	2014:2:17 00:00
Last tweet:	2014:2:19 23:49
Furthest of propagation (km):	1453.19
Speed of propagation (km/h):	37.77
Furthest of oldest tweet (km):	750.4
Speed from original (km/h):	11.64
Density estimation (tweets/km²):	0.000245
Area (km²):	3724389.62
Most message length:	137
Message length:	length > 10 = 888
	length > 50 = 710
	length > 100 = 550
Device info:	Longdo Traffic: 501
	iPhone: 196
	Android: 97
	foursquare: 63
	Instagram: 34
	iPad: 7
	Windows Phone: 6
	Android: 4
	TweetCaster for Android: 3
	Tweetbot for iOS: 1
	Tweetbot for Mac: 1

ในกรณีค้นหาด้วยหลายคำค้นหรือหลายคีย์เวิร์ดการแสดงผลการคำนวณจะเพิ่มคอลัมน์ที่ใช้ในส่วนของ Tweet Info เพื่อให้ง่ายต่อการเปรียบเทียบผลลัพธ์ ตามตัวอย่างรูปที่ 4.11 ซึ่งเป็นผลของการค้นคำว่า “vote no,no vote” เพื่อดูว่ามีผู้ใช้ทวิตเตอร์แสดงความคิดเห็นระหว่างการโหวตแบบไม่เลือกพรรคใดเลย (vote no) กับแบบไม่ไปเลือกเลย (no vote) ต่างกันอย่างไร ในที่นี้ผู้ทดสอบได้ทดสอบและเก็บผลลัพธ์ไว้ใน archive ทำให้จำนวนทวีตทั้งหมดมีเพียง 346 ข้อความซึ่งก็คือจำนวนทวีตที่เกี่ยวข้องกับคำค้นทั้งหมดนั้น จะเห็นว่าในจำนวน 346 ข้อความแบ่งเป็น vote no 113 ข้อความและ no vote อีก 233 ข้อความ โดยเริ่มปรากฏข้อความที่กล่าวถึงในช่วงเวลาไล่เลี่ยกันคือช่วง 8 โมงเช้าของวันที่ 1 กุมภาพันธ์ ค.ศ. 2014 หรือมองในแง่ของผู้ใช้งานจะเห็นว่ามีจำนวนผู้ใช้งานทวิตเตอร์ที่กล่าวถึงการไม่ไปเลือกตั้งมากกว่าไปเลือกแต่ไม่กาถึงสองเท่า และถูกพูดในวงกว้างถึง 1,550.33 กม. แต่มีการแพร่กระจายออกไปช้ากว่า 22.11 กม./ชม. และหากวัดกันจากทวีตที่เก่าที่สุดซึ่งอุปมาว่าเป็นทวีตเริ่มต้นของช่วงวันนั้นจะเห็นว่าถูกแพร่ไปไกลที่สุด 815.69 กม. สำหรับ no vote และ 512.56 กม. สำหรับ vote no ทั้งนี้การประมวลผลจะอยู่บนพื้นที่ขนาด 10,983,846.71 ตร.กม. และได้ความหนาแน่นของทวีตบนพื้นที่ต่างกัน 0.000011 ทวีตต่อตร.กม. จากที่กล่าวมาสามารถสรุปได้ว่าผู้ใช้ทวิตเตอร์ส่วนใหญ่ออกความคิดเห็นถึง no vote มากกว่า vote no และกระจายกันพูดออกไปไกลกว่าด้วย โดยผู้ใช้ส่วนใหญ่ทำการทวีตผ่านอุปกรณ์ iPhone ด้วยความยาวที่นิยมที่สุดคือ 7 ตัวอักษร (การวัดผลครั้งนี้ยังไม่ได้ทำการคัดกรองความยาวของภาษาจากทวิตเตอร์ทำให้ความยาวข้อความมีจำนวนน้อยกว่า 14 ตัวอักษรที่กำหนดเอาไว้)

Tweets Info

Information	Keyword: vote no	Keyword: no vote
Total # tweets:	113 / 346	233 / 346
Total # re-tweets:	0 / 0	0 / 0
Total # twitter users:	100	203
Oldest tweet:	2014:2:01 08:22	2014:2:01 08:09
Last tweet:	2014:2:03 00:28	2014:2:03 02:26
Furthest of propagation (km):	743.64	1550.33
Speed of propagation (km/h):	78.37	56.26
Furthest of oldest tweet (km):	521.56	815.69
Speed from original (km/h):	20.18	27.25
Density estimation (tweets/km²):	0.000010	0.000021
Area (km²):	10983846.71	10983846.71
Most message length:	19	7
Message length:	length > 10 = 110 length > 50 = 68 length > 100 = 36	length > 10 = 217 length > 50 = 130 length > 100 = 60
Device info:	iPhone: 51 Android: 41 Instagram: 8 foursquare: 5 Android: 3 WordPress.com: 2 TweetCaster for Android: 1 Tweetbot for iOS: 1 Windows Phone: 1	iPhone: 110 Android: 54 Instagram: 52 foursquare: 6 iPad: 5 WordPress.com: 2 TweetCaster for Android: 1 Android: 1 Windows Phone: 1 Path: 1

ภาพที่ 4.11 ข้อความทวีตบนกูเกิลเอิร์ธกรณีค้นด้วย “vote no,no vote”

4.2 การทดสอบ

การทดสอบระบบจะแบ่งออกเป็น 2 จุดประสงค์ คือ ตรวจสอบความน่าเชื่อถือ และการใช้งานของระบบเพื่อค้นดูพฤติกรรมที่เกิดขึ้น

4.2.1 การทดสอบความน่าเชื่อถือ

ทำเพื่อตรวจสอบความน่าเชื่อถือโดยทำการค้นหาทวีตบนพื้นที่จำกัดหนึ่งๆ ซึ่งมีข้อมูลเท็จจริงที่ใช้เปรียบเทียบได้ ได้แก่ ขนาดพื้นที่ และ การคำนวณระยะทาง เพราะส่วนของความหนาแน่นนั้นจะคำนวณด้วยจำนวนทวีตต่อพื้นที่ ดังนั้นความถูกต้องจะเกิดขึ้นเมื่อการคำนวณพื้นที่ถูกต้อง

4.2.1.1 การคำนวณระยะทาง

การคำนวณระยะทางขจัดหรือระยะทางที่สั้นที่สุดระหว่างพิกัดทางภูมิศาสตร์อันประกอบด้วยละติจูดและลองจิจูดสองจุดนั้น สมการ Haversine ถือว่าได้รับความนิยมมากที่สุด ในที่นี้จะทดลองคำนวณระยะทางผ่าน Haversine เทียบกับกับ Google Maps ซึ่งเป็น navigator tools ที่ได้รับความนิยม

ทดสอบวัดระยะทางเส้นอโศกมนตรีผ่านกูเกิลแมพส์ในรูปที่ 4.12 เส้นสีฟ้าคือเส้นทางเดินถนนซึ่งวัดระยะทางได้ 1.4 กม. เส้นสีแดงคือเส้นกระจัดที่คำนวณด้วยสมการ Haversine จะได้ระยะทางระหว่าง Phetchaburi MRT Station พิกัด 13.748870, 100.563323 กับถนน Asok Montri พิกัด 13.736655, 100.561199 จะได้ระยะทาง 1.378 กม. ดังนี้

แปลงพิกัดละติจูดและลองจิจูดให้อยู่ในหน่วยเรเดียน ซึ่งจะได้ [13.748870, 100.563323] องศา = [0.239963, 1.755161] เรเดียน และ [13.736655, 100.561199] องศา = [0.239750, 1.755124] เรเดียน

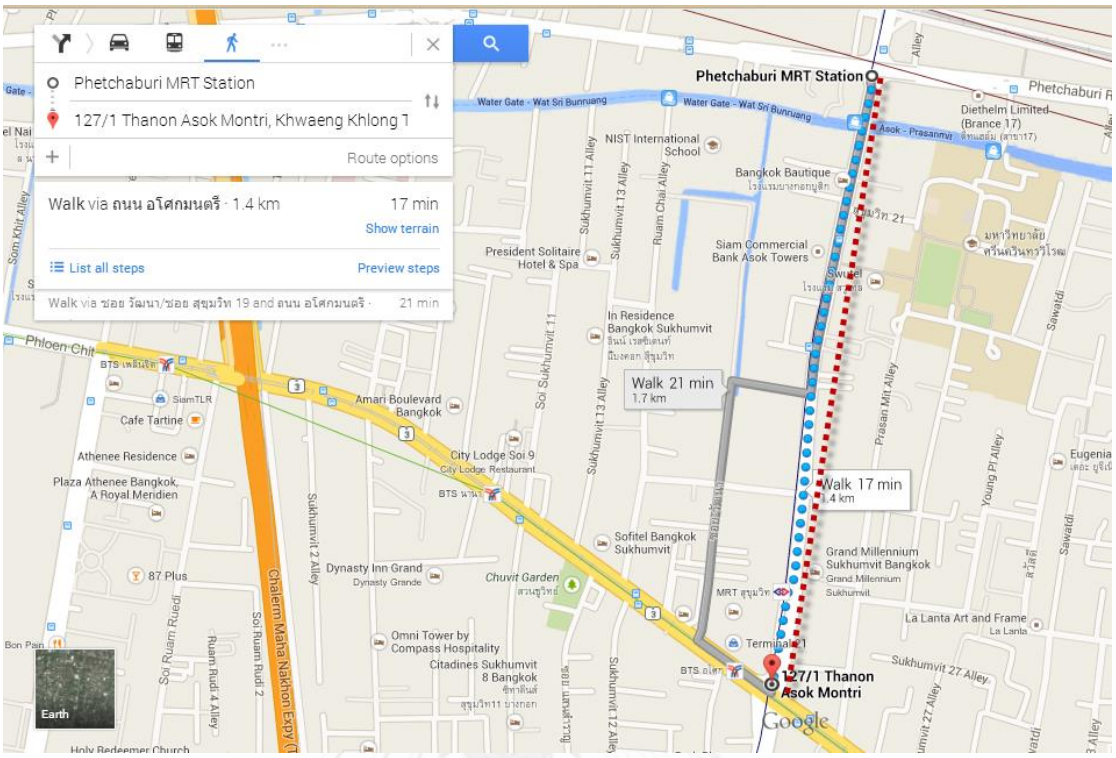
$$\Delta\lambda = \lambda_2 - \lambda_1 = -0.000037$$

$$\Delta\phi = \phi_2 - \phi_1 = -0.000213$$

$$haversin\left(\frac{d}{r}\right) = sin^2\left(\frac{\emptyset_2 - \emptyset_1}{2}\right) + cos(\emptyset_1)\,cos(\emptyset_2)\,sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)$$

$$h = sin^2\left(\frac{-0.000213}{2}\right) + cos(0.239963)\,cos(0.239750)\,sin^2\left(\frac{-0.000037}{2}\right)$$

$$d = 2(6{,}371)arcsin\left(\sqrt{h}\right) = 1.378$$



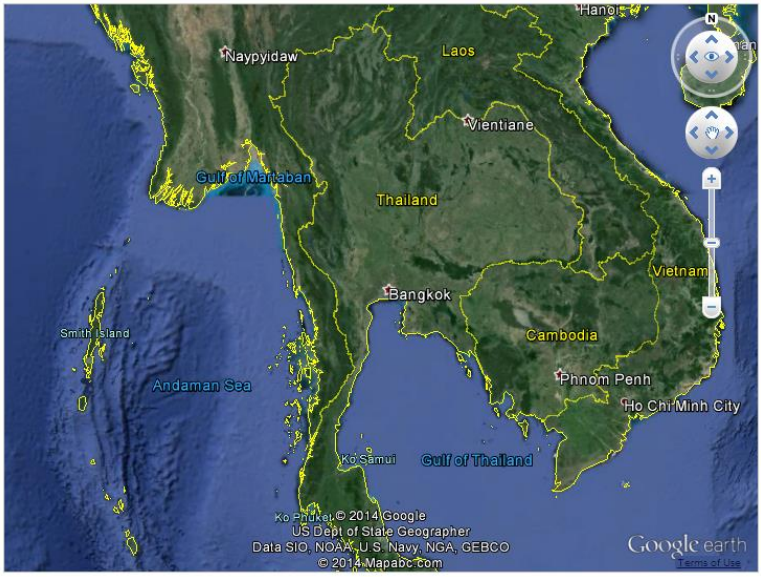
ภาพที่ 4.12 การวัดระยะทางถนนอโศกมนตรีด้วยกูเกิลแมพส์

4.2.1.2 การคำนวณพื้นที่

ในงานวิจัยจะอาศัยการคำนวณระยะทางที่หาด้วยสมการ Haversine มาคำนวณแบบพื้นที่สี่เหลี่ยมผืนผ้า หรือก็คือสูตร กว้าง x ยาว เนื่องจากพื้นโลกเป็นผิวโค้งพื้นที่ภายใน BBOX ที่เกิดขึ้นนั้นจะประกอบด้วยเส้นโค้งที่แนบไปกับพื้นโลกในลักษณะด้านตรงข้ามเท่ากัน เมื่อสามารถหาเส้นโค้งที่วางตัวตามผิวโลกได้แล้วจึงสามารถนำมาหาพื้นที่ด้วยวิธีพื้นฐานได้

ทั้งนี้ผู้วิจัยได้ทำการคำนวณและทดลองเปรียบเทียบขนาดพื้นที่ใน BBOX หนึ่งๆ โดย 2 วิธี คือวิธีคำนวณด้วยระยะทางที่เกิดจากสมการ Haversine กับการคำนวณพื้นที่ด้วย library ของกูเกิลแมพส์ (Google Maps) จะเห็นผลลัพธ์ดังรูปที่ 4.13 ซึ่งแสดงพื้นที่ของลานพุทธมณฑล ณ ความสูง 1,000 ม. จากพื้นทะเล จะเกิด error ขึ้นประมาณ 0.225% และเมื่อขยายภาพออกมาให้ความสูงอยู่ที่ระดับ 1,800,000 ม. จะมี error เกิดขึ้น 4.99% ณ จุดศูนย์กลางเดียวกัน ตามรูปที่ 4.14

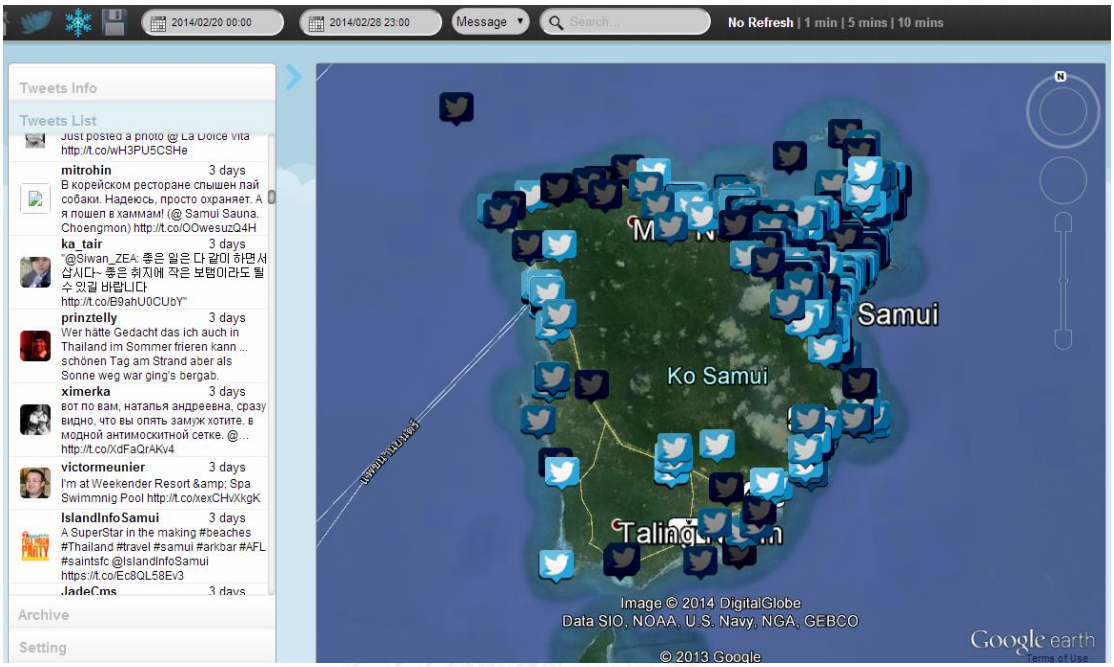
ภาพที่ 4.13 พื้นที่ลานพุทธมณฑล ณ ความสูง 1,000 เมตร



ภาพที่ 4.14 พื้นที่ประเทศไทยจุดศูนย์กลางที่ลานพุทธมณฑล ณ ความสูง 1,800,000 เมตร

4.2.2 การทดสอบการทำงานของระบบ

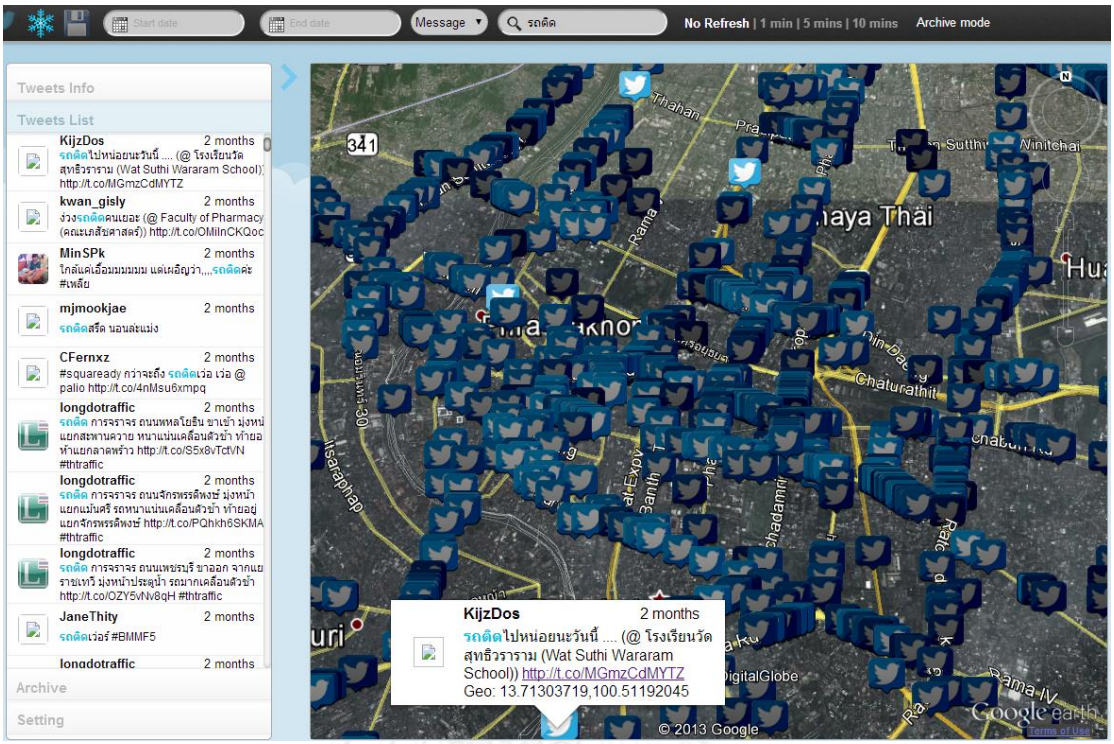
การทดสอบการทำงานของระบบนี้ช่วยให้เห็นพฤติกรรมบางอย่างของผู้ใช้ทวิตเตอร์และคุณลักษณะที่หลากหลาย ดังเช่นในรูปที่ 4.15 การเรียกค้นทวีตทั้งหมดในบริเวณเกาะสมุย ทวีตที่หนาแน่นโดยรอบตัวเกาะสมุยแสดงให้เห็นแหล่งที่อยู่อาศัยของชุมชนอย่างชัดเจน และเมื่อเทียบกับแผนที่ปกติจะพบว่าบริเวณที่มีทวีตหนาแน่นเป็นพิเศษนั้นเป็นส่วนเมืองของเกาะสมุย



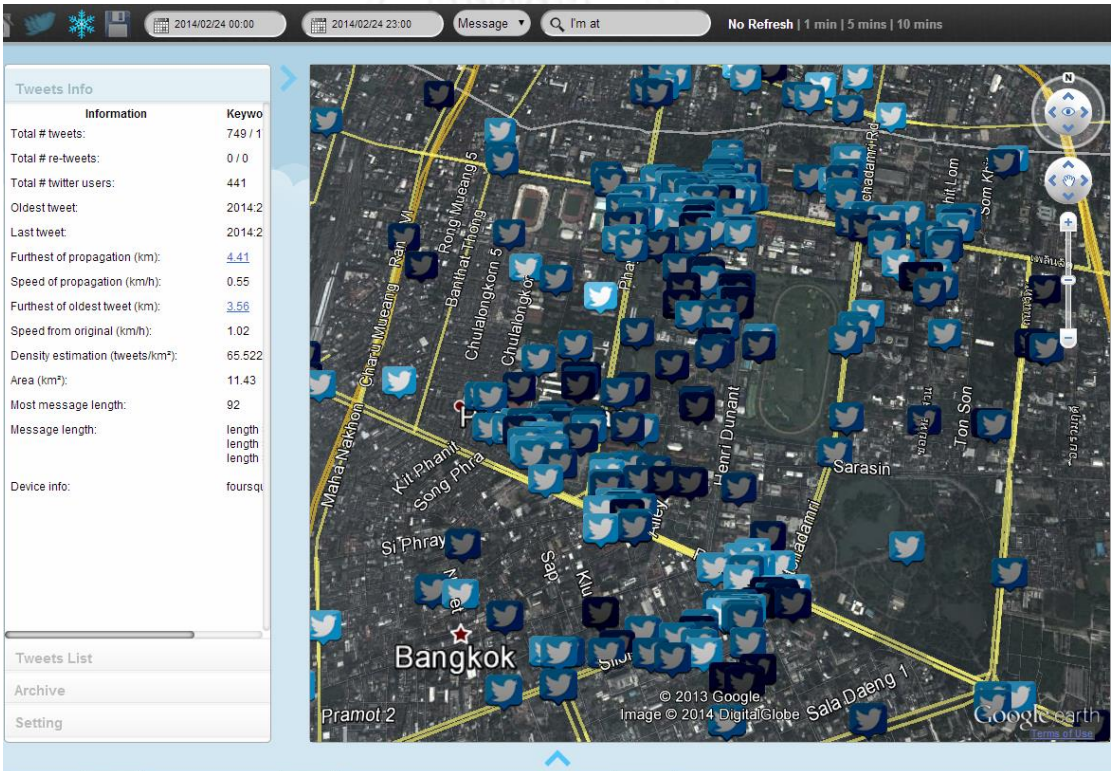
ภาพที่ 4.15 ผลลัพธ์จากการเรียกค้นทวีตทั้งหมดบริเวณเกาะสมุย

รูปที่ 4.16 เป็นทวีตเกี่ยวกับคำว่า “รถติด” การเรียงตัวของทวีตช่วยให้มองเห็นเส้นทางรถติดหรือเส้นทางหลักที่ผู้ใช้ทวิตเตอร์ใช้งาน และเมื่อพิจารณาจากสีของทวีตบนแผนที่จะพบว่ามีลักษณะสีเข้มเสียส่วนใหญ่ นั่นหมายถึงทวีตดังกล่าวเกิดขึ้นมาก่อนหน้าทวีตใหม่ๆ ซึ่งเป็นทวีตที่มีสีสว่างและปริมาณน้อย ช่วยให้คาดคะเนได้ว่า ณ ปัจจุบันภายในวันนี้หรือชั่วโมงนี้มีผู้ใช้ทวิตเตอร์พูดถึงคำว่า “รถติด” น้อยกว่าเมื่อวานหรือเมื่อชั่วโมงก่อนโดยไม่ต้องเปิดแผนภูมิเส้นเพื่อดูแนวโน้มของจำนวนทวีตรายวันหรือรายชั่วโมงเลย

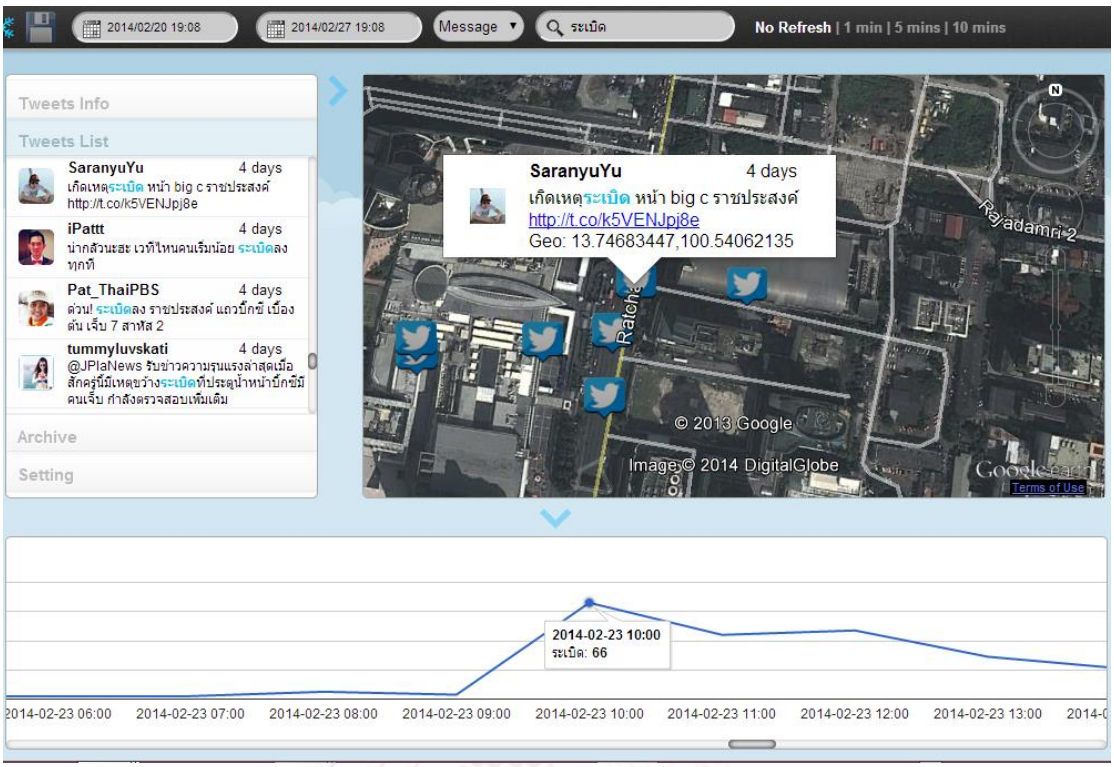
สำหรับรูปที่ 4.17 เป็นการ check-in สถานที่หนึ่งๆ ของผู้ใช้ทวิตเตอร์ ดังจะเห็นได้ว่าพื้นที่สาธารณะหรือแหล่งธุรกิจกระจุกตัวอยู่บริเวณใดบ้าง เพราะแม้ผู้ใช้ทวิตเตอร์จะอนุญาตให้มีการแนบพิกัดบนข้อความของตนได้ แต่ส่วนมากจะคำนึงถึงความเป็นส่วนตัวของตนจึงหลีกเลี่ยงการโพสข้อความที่แนบพิกัดที่พักอาศัย รูปที่ 4.18 เป็นข่าวการระเบิดในวันที่ 23 กุมภาพันธ์ ค.ศ. 2014 บริเวณใกล้เคียงกับจุดเกิดเหตุผู้ใช้ทวิตเตอร์จะมีการแจ้งเตือนข่าวสารให้กับคนอื่นๆ ได้ทราบผ่านการทวีต ซึ่งเห็นได้จากปริมาณที่เพิ่มขึ้น ณ ขณะหนึ่งว่ามีผู้ใช้ทวิตเตอร์พูดถึงระเบิดอย่างกระทันหัน



ภาพที่ 4.16 ผลลัพธ์การค้นหาด้วยคำว่า “รถติด”

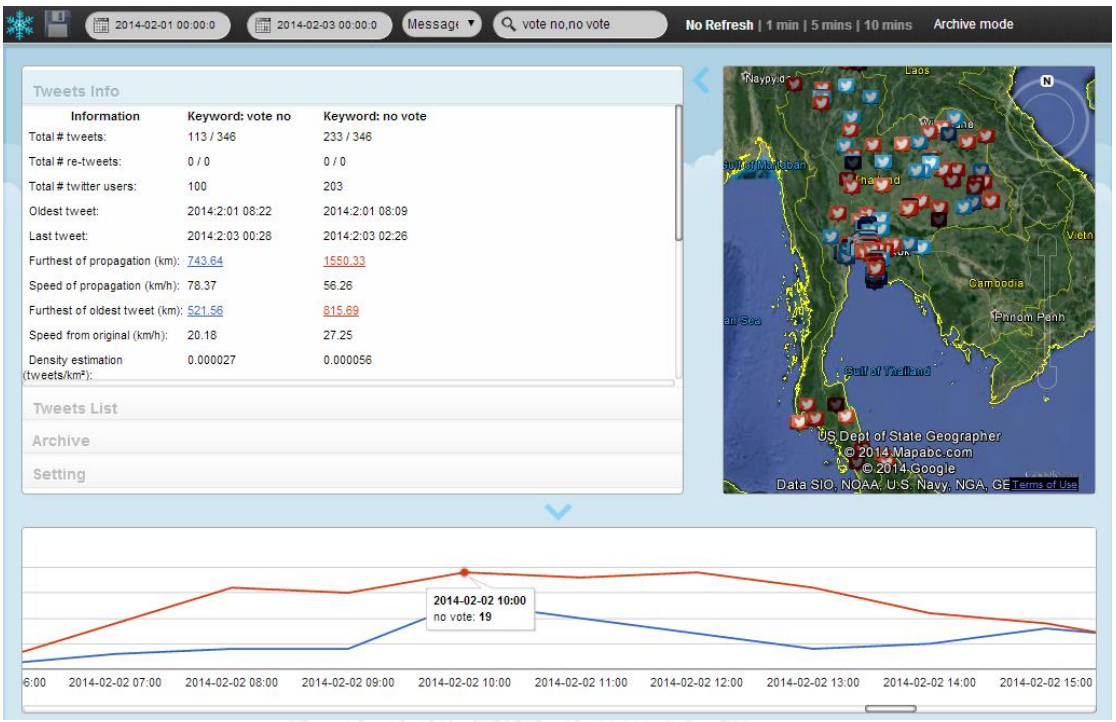


ภาพที่ 4.17 ผลลัพธ์การค้นหาด้วยคำว่า “I’m at”



ภาพที่ 4.18 ผลลัพธ์การค้นหาด้วยคำว่า “ระเบิด”

การค้นหาด้วยหลายคำค้น “vote no, no vote” ตามรูปที่ 4.19 เป็นการเปรียบเทียบข้อคิดเห็นระหว่างการไปเลือกตั้งแต่ไม่กาเลือกพรรคใดกับการไม่ไปเลือกตั้งเลย เพื่อดูว่าผู้ใช้ทวิตเตอร์ให้ความสนใจกับแบบใดมากกว่ากัน ซึ่งผลจากการค้นหาจะทำให้เห็นได้อย่างชัดเจนว่าผู้ใช้ทวิตเตอร์กล่าวถึงคำว่า no vote มากกว่าเกือบเท่าตัว และมีข้อมูลกระจายออกไปทั่วประเทศ โดยเทียบเป็นตัวเลขจะเห็นได้ชัดเจนขึ้นว่าระยะทางที่ไกลที่สุดระหว่างทวีตที่เกิดขึ้นชของ no vote นั้น (1,550 กิโลเมตร) ไกลกว่าของข้อความที่พูดถึง vote no (743 กิโลเมตร)



ภาพที่ 4.19 ผลลัพธ์การค้นหาด้วยคำว่า “vote no, no vote”

บทที่ 5
บทสรุป

5.1 สรุปผลการวิจัย

งานวิจัยนี้ได้พัฒนาเว็บแอพพลิเคชั่นเพื่อนำเสนอการวิเคราะห์และการแสดงผลของชุดข้อมูลจากทวิตเตอร์ในเชิงภูมิศาสตร์และเวลา ช่วยให้สามารถศึกษาคุณลักษณะหรือพฤติกรรมของผู้ใช้งานทวิตเตอร์ได้หลากหลาย อย่างการใช้งานสื่อออนไลน์ผู้ใช้ทวิตเตอร์อาจไม่ได้ทวีตข้อความนั้นๆโดยตรงจากโปรแกรมทวิตเตอร์เอง แต่อาจเชื่อมโยงมาจากแอพพลิเคชั่นอื่นๆเช่น Foursquare, Instagram, Path ซึ่งหมายความว่าผู้ใช้งานจำนวนไม่น้อยที่เป็นสมาชิกและเสพข่าวจากหลายๆ แหล่ง ทำให้เวลานำข้อความเหล่านั้นมาปักหมุดบนแผนที่มีลักษณะกระจุกตัวอยู่ ณ จุดเดียวกันของพิกัด การทวีตข้อความสำหรับประเด็นร้อนในข่าวแต่ละวันก็เป็นอีกหนึ่งพฤติกรรมที่ทำให้เห็นการเปลี่ยนแปลงที่ชัดเจน จำนวนทวีตจะมีปริมาณสูงขึ้นอย่างรวดเร็วและค่อยๆลดปริมาณลงเมื่อข่าวเริ่มเก่าขึ้น และจะกลับมาเพิ่มอีกครั้งเมื่อมีเหตุการณ์ใดๆ มากระตุ้นให้เกิดความสนใจ ทั้งนี้ความหนาแน่นของทวีตยังช่วยบอกให้ทราบว่าหัวข้อดังกล่าวได้รับความสนใจมากในพื้นที่ใด หรือแม้กระทั่งการเคลื่อนตัวของข่าวก็ได้เช่นกัน

แม้ว่างานวิจัยจะช่วยเสริมให้สามารถศึกษาพฤติกรรมของผู้ใช้ทวิตเตอร์ได้มากยิ่งขึ้น แต่ด้วยจำนวนทวีตที่มีพิกัดแนบมาด้วยมีประมาณน้อยกว่าจำนวนทวีตทั้งหมดที่เกิดขึ้น คือประมาณ 10 - 15% ของทวีตทั้งหมดในประเทศไทย ดังนั้นผลลัพธ์ที่ได้จากการวิเคราะห์นี้จึงไม่เหมาะสมในการนำมาตัดสินพฤติกรรมของผู้ใช้ทวิตเตอร์ทั้งหมดเพียงแต่เป็นข้อมูลประกอบเท่านั้น และเมื่อเทียบจำนวนผู้ใช้ทวิตเตอร์ที่ใช้งานจริงของประเทศไทยจะมีประมาณ 200,000 ผู้ใช้งาน ซึ่งเทียบกับจำนวนประชากรทั้งหมด 64 ล้านคน [34] แล้วนับว่าเป็นกลุ่มคนที่น้อยมากจึงอาจยังไม่เห็นประสิทธิภาพมากพอสำหรับการใช้สังเกตพฤติกรรมอื่นๆ ในประเทศไทย ผู้พัฒนาเห็นควรว่าระบบนี้เหมาะสมกับประเทศหรือพื้นที่ใดๆ ที่ประชากรนิยมใช้ทวิตเตอร์เป็นสื่อกลางในการสื่อสารหลัก

นอกจากนี้ยังพบว่าข้อความในประเทศไทยที่รับได้จากทวิตเตอร์แล้วนำมาเก็บในฐานข้อมูลนั้น จะมีขนาดประมาณ 50 MB ต่อวัน ทำให้ระบบจะต้องจัดเก็บข้อมูลประมาณ 350 MB ต่ออาทิตย์และอาจมีขนาดสูงขึ้นหากในอนาคตทำการขยายพื้นที่ในการขอรับข้อความจากทวิตเตอร์ ซึ่งจะมีผลกระทบต่อประสิทธิภาพในการทำงานของระบบให้ต่ำลง ณ การทดลองนี้พบว่าระบบสามารถ

ตอบสนองต่อผู้ใช้งานได้ภายใน 0.01 วินาทีต่อทวีต (ทดลองด้วย SAMSUNG Series5) และข้อความ JSON ที่ส่งคืนให้แก่ไคลเอนต์จะมีขนาดประมาณ 120 B ต่อทวีตหลังทำการบีบอัดแล้ว

จากการวิจัยที่ผ่านมาผู้วิจัยยังพบปัญหาบางอย่างซึ่งสมควรได้รับการแก้ไขหรือหาแนวคิดเพื่อแก้ปัญหาในอนาคตซึ่งจะขอกล่าวในหัวข้อถัดไป แต่โดยสรุปแล้วการพัฒนาแอพพลิเคชั่นนี้ขึ้นมาทำให้ได้พบกับอีกหนึ่งแนวทางในการศึกษาพฤติกรรมของผู้ใช้ทวิตเตอร์

5.2 ปัญหาและข้อจำกัดที่พบจากการวิจัย

5.2.1 ปัญหาประสิทธิภาพของระบบเมื่อข้อมูลมีขนาดใหญ่ขึ้น

5.2.2 ข้อความรีทวีตซึ่งเป็นข้อความที่ผู้ใช้ทวิตเตอร์ทำการส่งต่อไปยังหน้าเพจของตัวเอง ทำให้เกิดการกระจายข้อมูลออกไปในวงกว้างนั้นยังไม่สนับสนุนการแนบพิกัดลงบนข้อความที่รีทวีต ดังนั้นในงานวิจัยนี้จึงไม่สามารถวิเคราะห์ข้อมูลดังกล่าวได้

5.2.3 มีข้อความปริมาณมากที่ผู้ใช้งานไม่อนุญาตให้ระบุพิกัดที่อยู่ไปกับเนื้อข้อความทำให้ระบบไม่สามารถรับมาประมวลผลได้

5.2.4 ระบบนี้ถูกออกแบบให้การวิเคราะห์ส่วนใหญ่อยู่ที่ฝั่งไคลเอนต์ดังนั้นจึงอาจทำให้ฝั่งไคลเอนต์อาจต้องการคอมพิวเตอร์ที่มีประสิทธิภาพระดับหนึ่งในการเข้าใช้งาน ซึ่งขณะนี้ผู้ทำวิจัยใช้คอมพิวเตอร์ SAMSUNG รุ่น 530U4B-S02 (Series 5) หน่วยปฏิบัติการวินโดว์ Corei5 RAM ขนาด 6 GB และเบราว์เซอร์ Google Chrome version 33.0.1750.154 m ในการทดสอบระบบ

5.3 ข้อเสนอแนะ

จากการทดลองและทดสอบหลายๆ ครั้งด้วยเงื่อนไขที่แตกต่างกันไป จึงพบปัญหาและข้อจำกัดในการทำวิจัยดังข้างต้น ซึ่งการแก้ไขสามารถทำได้ด้วยงานวิจัยในอนาคต เช่น

5.3.1 การปรับปรุงประสิทธิภาพในการทำงานของระบบ

- การจัดทำฐานข้อมูลให้แบ่งการจัดเก็บเป็น Partition โดยอาจจัดทำเป็นช่วงเวลารายวัน หรือจัดเก็บตามโซนของพื้นที่ก็ได้เช่นกัน ทั้งนี้จะทำให้การค้นหาเป็นไปได้รวดเร็วยิ่งขึ้นเมื่อปริมาณข้อความทวีตในระบบมีมาก
- ออกแบบให้ส่วนการวิเคราะห์ข้อมูลมาอยู่ทางฝั่งเซิร์ฟเวอร์บางส่วน เพื่อลดภาระของไคลเอนต์และเปิดโอกาสให้ผู้สนใจใช้เว็บ WOT สามารถใช้งานได้อย่างมีประสิทธิภาพมากขึ้น

5.3.2 ข้อจำกัดในส่วนของข้อความรีทวีตที่ไม่มี Geolocation หรือพิกัดทางภูมิศาสตร์แนบเข้าไปด้วย ทำให้การเรียกคืนข้อความจากทวิตเตอร์นั้น ไม่นับรวมข้อความรีทวีตเข้ามา ซึ่งหากต้องการเรียกคืนข้อมูลส่วนนี้อาจต้องเพิ่มช่องทางหรือ session เพื่อเรียกคืนทวีต ซึ่งจะต้องทำการติดต่อกับทวิตเตอร์ด้วยรหัสที่ต่างกันกับ session หลัก และกรองด้วย track parameter ที่ใช้ร้องขอข้อความจากทวิตเตอร์ด้วยคีย์เวิร์ดชุดหนึ่งแทน ในที่นี้อาจจะกำหนดค่าเป็น track='RT' ก็ได้

5.3.3 ต่อเนื่องจากข้อ 5.3.2 ที่ข้อความรีทวีตจะไม่ถูกแนบพิกัดมาด้วย ทำให้ไม่สามารถวิเคราะห์ข้อมูลเหล่านั้นในเชิงพื้นที่ได้ อาจใช้วิธีโดยอ้อมเพื่อค้นหาพิกัดที่เป็นไปได้ ดังเช่น การค้นหาพิกัดล่าสุดของผู้ใช้จากฐานข้อมูล หรือการใช้ Home town หรือ timezone เป็นการอ้างอิงก็ทำได้เช่นกัน เพียงแต่ความหมายที่สื่อออกมาจะเปลี่ยนไปเล็กน้อย เพราะทวีตที่แสดงขึ้นมาด้วยที่อยู่ที่ไม่ถูกต้องนี้ ไม่อาจบอกได้ว่าข่าวสารที่รีทวีตออกไปนั้นแพร่กระจายไปถึงในประเทศใดบ้าง

5.3.4 การค้นหาเนื้อความที่เกี่ยวข้องสำหรับงานวิจัยชิ้นนี้ไม่ได้รองรับการค้นหาด้วยความหมายเอาไว้ ทำให้ผลลัพธ์ที่ออกมาอาจยังไม่เหมาะสมในการนำไปใช้ต่อ หากสามารถผนวกเอาการค้นหาแบบ Semantic หรือใช้ ontology เฉพาะเรื่องใดเรื่องหนึ่งในการเก็บข้อมูลและแสดงผล ระบบก็จะมีประโยชน์และเหมาะสมในการนำไปใช้มากยิ่งขึ้น

รายการอ้างอิง

1. *Twitter*. March 2007; Available from: http://en.wikipedia.org/wiki/Twitter.
2. Millward, S. *Thailand Now Has 18 Million Social Media Users*. May 2013; Available from: http://www.techinasia.com/thailand-18-million-social-media-users-in-2013.
3. Ming, H., et al. *Visual sentiment analysis on twitter data streams*. in *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*. 2011.
4. Soler, J.M., F. Cuartero, and M. Roblizo. *Twitter as a Tool for Predicting Elections Results*. in *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on*. 2012.
5. Sakaki, T., M. Okazaki, and Y. Matsuo. *Earthquake shakes Twitter users: real-time event detection by social sensors*. in *Proceedings of the 19th international conference on World wide web*. 2010. ACM.
6. Nation, T. *Twitter the most popular channel: ZocialEye.com*. 2013 November 2013; Available from: http://www.nationmultimedia.com/politics/Twitter-the-most-popular-channel-ZocialEye-com-30219481.html.
7. Tuetrakul, J., *Social Network on Google Earth*. 2011, Chulalongkorn University.
8. Vithyaviranont, V. and V. Muangsin, *Spatial-Temporal Analysis and Visualization of Twitter Data on Google Earth.* The 10th National Conference on Computing and Information Technology, 2014.
9. Twitter, I. *Learn the basics*. 2014 [cited 2014]; Available from: https://discover.twitter.com/learn-more.
10. Twitter, I. *Twitter API Documentations*. 2014 [cited 2014 April]; Available from: https://dev.twitter.com/docs.
11. BrightPlanet. *TWITTER FIREHOSE VS. TWITTER API: WHAT'S THE DIFFERENCE AND WHY SHOULD YOU CARE*. 2013 [cited March 2014]; Available from: http://www.brightplanet.com/2013/06/twitter-firehose-vs-twitter-api-whats-the-difference-and-why-should-you-care/.
12. Twitter, I. *Public streams*. 2013 [cited March 2014]; Available from: https://dev.twitter.com/docs/streaming-apis/streams/public.
13. Developers, G. *Geolocation*. 2013 [cited March 2014]; Available from: https://developers.google.com/maps/articles/geolocation
14. Twitter, I. *Streaming API request parameters*. 2013 [cited October 2013]; Available from: https://dev.twitter.com/docs/streaming-apis/parameters#locations
15. Twitter, I. *Adding your location to a Tweet*. [cited April 2013]; Available from: http://support.twitter.com/articles/122236#.

16. Roesslein, J. *Tweepy v1.4 Documentation*. 2009 [cited 2013]; Available from: http://pythonhosted.org/tweepy/html/getting_started.html.
17. ศุภอรรถกร, ช., จัดการฐานข้อมูลด้วย *MySQL*. 2555, Simplify.
18. Sinnott, R.W., *Virtues of the Haversine.* Sky and telescope, 1984. **68**: p. 158.
19. Google. *Google Earth*. 2005 April 2013]; Available from: http://www.google.com/earth/index.html.
20. Google. *Google Earth API Developer's Guide*. 2013 [cited September 2013]; Available from: https://developers.google.com/earth/documentation/.
21. Google. *Keyhole Markup Language*. 2013 [cited November 2013]; Available from: https://developers.google.com/kml/documentation/kml_tut
22. Google. *Google Charts*. 2012 [cited 2013]; Available from: https://developers.google.com/chart/.
23. *JavaScript*. [cited 2014]; Available from: http://en.wikipedia.org/wiki/JavaScript.
24. *Introducing JSON*. [cited 2014]; Available from: http://www.json.org/.
25. *IndexedDB*. [cited April 2014]; Available from: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API.
26. Demirbas, M., et al. *Crowd-sourced sensing and collaboration using twitter*. in *World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a*. 2010. IEEE.
27. Doan, S., L. Ohno-Machado, and N. Collier. *Enhancing Twitter Data Analysis with Simple Semantic Filtering: Example in Tracking Influenza-Like Illnesses*. in *Healthcare Informatics, Imaging and Systems Biology (HISB), 2012 IEEE Second International Conference on*. 2012. IEEE.
28. Phuvipadawat, S. and T. Murata. *Breaking news detection and tracking in twitter*. in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*. 2010. IEEE.
29. Chatfield, A.T. and U. Brajawidagda. *Twitter Early Tsunami Warning System: A Case Study in Indonesia's Natural Disaster Management*. in *System Sciences (HICSS), 2013 46th Hawaii International Conference on*. 2013. IEEE.
30. Esmin, A.A.A., R.L. de Oliveira, and S. Matwin. *Hierarchical Classification Approach to Emotion Recognition in Twitter*. in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*. 2012.
31. Twitter, I. *Location, Location, Location*. 2009 [cited 2013]; Available from: https://blog.twitter.com/2009/location-location-location.

32. Hansu, G., et al. *Fusing Text and Frienships for Location Inference in Online Social Networks*. in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on*. 2012.

33. Weidemann, C. and J. Swift, *Social Media Location Intelligence: The Next Privacy Battle - An ArcGIS add-in and Analysis of Geospatial Data Collected from Twitter.com.* International Journal of Geoinformatics, 2013. **9**: p. 21-27.

34. สถาบันวิจัยประชากรและสังคม. ข้อมูลประชากรในประเทศไทย. [cited April 2014]; Available from: http://www.ipsr.mahidol.ac.th/ipsr-th/population_thai.html.

ภาคผนวก

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวอิสราภรณ์ วิทยวิรานนท์ เกิดเมื่อวันที่ 14 กรกฎาคม พ.ศ. 2530 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาระดับปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศกรรมโทรคมนาคม เกียรตินิยมอันดับ 1 จากคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2552 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ณ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2554 งานวิจัยที่สนใจ ได้แก่ เว็บเซอร์วิซ การสังเคราะห์ข้อมูลขนาดใหญ่จากสื่อสังคมออนไลน์