THE UNIVERSITY OF WESTERN AUSTRALIA

Achieve International Excellence

School of Electrical, Electronic & Computer Engineering

RESEARCH PROJECT

Semester 1, 2019

Identifying Useful Features for Detecting False Information in Social Media

Author: **Hao Sun 22199907**
Supervisor: **Dr. Jin Hong**

16 May, 2019

Abstract

Information credibility has become an increasing concern for online social media. Due to the characteristics of online information propagation, which is massive, fast and widespread, it is almost impossible to manually detect the spread of false information in social media. Therefore, the application of machine learning in online false information detection has been the prevalent method to address this issue. Existing research has shown that the performance of the ML model can approach human level with reasonable features and hyperparameters. Supervised classification algorithms are usually ideal solutions. However, the accuracy of existing solutions to detect and identify false information in social media is impractical, mainly due to the difficulty of identifying useful features to use.

This project proposes an automated false information (e.g. bad rumour) detection system on Twitter dataset using machine learning (ML) algorithms with a large number of features related to social media contents. To validate our results, a publicly available twitter dataset PHEME was used, which contains the spread of false information among users. First, feature analysis and extraction modules are implemented, which results in an over 2000-dimensional feature matrix containing user-based, content-based and propagation-based features. Second, six supervised ML models under scikit-learn library are implemented, including Logistic regression, Decision Tree, KNN (K nearest neighbours), LDA (Linear discriminant analysis), GNB (Gaussian Naïve Bayes), and SVM (Support Vector Machine). And a Gaussian Naïve Bayes classifier developed by myself is also applied. The result shows that most ML models achieve a weighted average F1 score between 0.8 and 0.9, which is a significant improvement over previous work which only achieved 0.61. Logistic regression under scikit-learn is the overall best model. Additionally, a sensitivity analysis is carried out to investigate the effect of adjusting the training set size and the result indicates that a larger proportion of data used for training the ML models achieves higher accuracy. Moreover, the rumor detection system is applied to various rumour event datasets and it turns out the proposed system remains robust to the change of the rumour topic. Lastly, the system is tested with new rumour events on different feature sets. Content-based features outperforms user and propagation based features and GNB achieves the best FN rate.

Contents

1. Introduction

In today's big data era, enormous data are created, stored, propagated and analysed every day. For instance, Twitter is one of the largest worldwide online social media platforms, which generates millions of message (named *tweets*) every day. As of 2016, it had more than 319 million monthly active users [1]. Twitter users are allowed to post up to 140-character-long tweets via the website interface or the mobile app anytime, which makes it possible that massive information can reach a huge group of users quickly. So it provides an ideal platform for the spread of breaking news. Take 2016 U.S. presidential election for example, where 40 million election related tweets had been sent by 10pm on the Election Day [2]. However, Twitter also gives malicious users (i.e. rumour spreaders) a platform to disseminate false information (e.g. bad rumours). Due to the lack of means to validate the factual correctness of social media contents easily, people may not be able to distinguish between true and false information. As time passes, false information is deliberately created to be more convincing, deceiving even the trained readers [3]. Researchers found that after the 2010 earthquake measuring 8.8 on the Richter scale in Chile, many rumours were propagated on Twitter due to the lack of trustworthy sources, which caused panic and chaos in the local society [4]. Unfortunately, bad rumours are spreading among social network more prevalently, and sometimes they can have a severely negative impact on real world.

False information has gained much attention in social media recently. A number of researchers have analysed the characteristics of them and proposed several feasible strategies to detect and mitigate their influence in the early phase. Traditionally, people tend to manually detect rumours by comparing against a solid source such as the rumour debunking website FactCheck.org. These tasks are usually assigned to several professional analysts, and they would examine and annotate the credibility of the message. The downside of such method is that it is too slow and laborious, not to mention error prone. So, it cannot effectively cope with the explosive emerging and diversely changing rumours in the early phases its spread. It may take few weeks or even months to identify and annotate all rumours, but by the time it is usually too late since the rumours have already been widely spread and caused damage. Rather than manually tagging rumours in social media, many cutting edge techniques and approaches have been developed to improve the performance. Machine learning (ML) is a new trend for many data classification and prediction problems. Specifically, the one can extract features or attributes of each example that are to be classified as rumour or non-rumour from the original dataset. Then, we can feed the features to the ML model, and the learned classifier is able to predict rumours from non-rumours automatically. Many ML-based rumour detection models achieved competitive performance, but sometimes over-fitting or under-fitting the model can cause an issue due to the complexity of features or the lack of training examples. Another new approach proposed by Zhao et al. [5] used the "enquiry tweets" that question a

controversial rumour topic. The “enquiry tweets” are clustered and extracted based on a manually designed regular expression. Then, the property of tweets clusters can be analysed, and each tweet within a cluster is deemed to be rumour or non-rumour according to the appearance of challenging inclination of that tweet.

An effective rumour detection system should remind users about the credibility of the information, or if it might be proven to be a rumour later. This does not only reduces the probability of ambiguous information that could eventually turn out to be a rumour and end up being widely spread, but also reduces negative influence on individuals and online environment [6]. In this work, an automated rumour detection system is developed for Twitter using ML models, which is able to distinguish between tweets identified as rumours and non-rumour tweets. With the generated results, the Twitter operator could take further actions (e.g. block inauthentic tweets or notify users the credibility of a tweet) to prevent the propagation of bad rumours.

This project aims to identify social media features that can be used to detect bad rumours in order to improve the accuracy of detection. A new feature set is developed, combining some of the previous features together. To evaluate the accuracy of the ML models to detect bad rumours in Twitter, a Twitter dataset named PHEME has been used [7], which is a structured dataset with classified bad rumours. The PHEME dataset contains over 5000 tweets across 5 rumour topics. In a previous work using the same dataset, they achieved the F1 score of 0.61 [7], where most of the ML models using the new feature set in this work achieved the value ranging from 0.8 to 0.9, demonstrating a significant improvement. All the implementation done for the project can be found in the Github repository. [34]

The contributions of this project are as follows.

1. Identify useful set of features and their combinations for detecting bad rumours in social media platforms using different ML models;
2. Implement six ML models under scikit-learn and design a GNB model from scratch and evaluate the accuracy of detection bad rumours using the PHEME dataset;
3. Conduct experimental analysis to evaluate the performance and accuracy of different ML models, training set sizes, datasets, feature sets and new rumour events.

The rest of the report is organised as follows. Section 2 presents the related work. Section 3 outlines the pipeline of this project and the details of false information detection in social media, focusing on the dataset analysis, feature extraction and ML model establishment. Section 4 presents the performance evaluation of using different ML models to detect bad rumours, performance against different rumour events and features, performance of testing new rumours. In section 5, it is about discussions on key hypothesis and future work. Finally, section 6 concludes this project and outlines the critical results of experiments.

2. Related Work

2.1 Rumour identification and analysis

2.1.1 Rumour definition

The definition of rumour can vary between different literatures. But most of them are in line with the Cambridge English Dictionary which defined rumour as “unofficial, interesting story or piece of news that might be true or invented, and that is communicated quickly from person to person”. Qazvinian et al. [8] define a rumour as “a statement whose truth-value is unverifiable or deliberately false”. DiFonzo et al. [9] define rumours as “unverified and instrumentally relevant information statements in circulation that arise in contexts of ambiguity, danger or potential threat, and that function to help people make sense and manage risk". Rumours that are invented falling into the false information category, especially ones that negatively impact persons, groups, societies etc. Kumar et al. [11] point out rumours on social media could manipulate stock markets and create panic in disasters and terrorist attacks. Research [22] shows rumours usually spread widely and deeply and have a high risk of becoming viral during the initial stage (within 12 hours before the debunking information emerge). Sometimes the invalid information can even be disseminated by reputed news organizations which cause substantial impact on society and local population. [23] Hence, it is of paramount importance to detect bad rumours.

2.1.2 False information category

In 2018, Savvas et. al. [10] carried out a research regarding to the false information on online social networks. They classified the false information to ten types based on content and clustered them into three groups based on their severity. Table 1 lists the false information category:

Table 1: False information category[10]

Fake News	Biased/Inaccurate News	Misleading/Ambiguous News
Fabricated	Hoaxes	Rumours
Propaganda	Hyperpartisan	Clickbait
Imposter	Fallacy	Satire News
Conspiracy Theories	-	-

Kumar and Shah [11] classified false information based on the intent and knowledge of false information. In intent aspect, if the information is created without the intent to mislead others due to the lack of understanding or unconscious distortion of source information, it is defined as misinformation. Otherwise, it is called disinformation. On the other hand, false information can be divided into opinion based and fact based by the knowledge. Opinion based one refers to the dishonesty of individuals when they express their reviews towards a product or service. Fact based one refers to

information that conflict with ground truth. Table 2 illustrates the category defined by them.

Table 2: False information category [11]

Categorised by intent		Categorised by knowledge	
Misinformation	Disinformation	Opinion-based	Fact-based
e.g. Urban Legend	e.g. Fake News	e.g. Movie Reviews	e.g. Rumours

2.1.3 Characteristic of rumours

Many studies analysed the textual and propagation characteristic of false information. These characteristics can be indicative for rumour detection and directly related to the quality of designed features. Horne and Adali [12] found fake news tend to emphasize the title by using words or phrases that easy to understand and fewer stop words. The articles are usually concise, repetitive and with fewer technical words. Bessi et al. [13] explored the twitter dataset during presidential election of the United States, and they found around 20% of tweets are generated by automated fake account in a short time period. Azaria et al. [14] discovered that some Twitter fake accounts tend to have close following relationships between each other. They form a number of overlapped bot account follower-followee networks. This finding can be exploited to capture a group of associated fake accounts once a fake account has been identified. Kumar and Shah [11] argue that rumour in social media can be disseminated much faster, broader and more likely to reach deep users in a following network compared to real posts.

2.2 Twitter data retrieval

There are two main approaches to obtain Twitter data. The first one is using publicly available Twitter rumour dataset. The data of existed dataset usually have already been developed by other researchers and annotated by journalists. The other one is directly mining Twitter data through Twitter API. If a rumour event is already known to be unreal, we can design a sequence of key words which is also called regular expression queries to cover the main point of a rumour and send them to Twitter API to retrieve tweets. For example, if a rumour is “Obama has a long hair.” The regular expression query can be (“Obama” | “US president”) & (“hair” | “haircut”). However, this method can be constrained by the speed limitation imposed by Twitter API and it may take a long time to do data annotation. Another downside is the design of the key words is entirely relied on human intelligence which may not cover all patterns of rumour tweets and result in a low recall. Moreover, the key words cannot adapt to the variation of rumours due to the change of time. By contrast, the advantage is that we can extract twitter topics that we are interested in.

Horne and Adali [12] used Buzzfeed election dataset and Burfoot and Baldwin dataset which have been developed in other researchers work for fake news detection. They also created their own political news dataset containing 75 stories across three types

of news namely real, fake, and satire. All the information is obtained from valid online source such as snopes.com. Qazvinian et al. [8] designed regular expression queries to match the content of each tweet via Twitter API. Therefore, rumour related tweets can be extracted. Then they employed two annotators to check the credibility of each tweet according to About.com's Urban Legends reference site. Rumours are labeled as 1 and non-rumours as 0. Castillo et al. [15] focused on the Twitter rumour topics detected by Twitter Monitor, a system that can output a sequence of key words that appeared frequently in the tweets during the past few days. Then they extracted tweets that matching the key words. 2500 rumour topics were delivered to the evaluators of Amazon's Mechanical Turk for annotation. They called it "human intelligence task" or HIT.

2.3 Feature analysis

Features are the representative of raw data and the direct input of machine learning models. Well-designed feature set can improve performance significantly. Current literatures in rumour detection on Twitter not only focus on lexical features that extracted from content, but also involve propagation features extracted from information flowing network and Twitter specific features such as hashtags and URLs. Castillo et al. [15] proposed 68 features that can be grouped into four categories namely message, user, topic, and propagation. They implemented a best-first selection method and selected 15 most important features including the number of status of users, the number of followers and friends of users, if the tweets contain URLs, user mentions and question marks etc. They found the features derived from sentiment analysis also play an important role. Buntain et al. [16] analysed 45 features across four groups: structural, user, content and temporal. They adopted most of the 15 best features described in Castillo et al.'s work except for two features on most frequent web links. Yang et al. [17] developed their feature set based on content, client, account, location and propagation of information on Sina Weibo. They proposed two new features that no previous works had ever investigated which are the client program used for posting and the location where user post the information. Zubiaga et al. [7] applied social features and content-based features. They used the word vector to represent each tweet and used part-of-speech (POS) features to represent the role that each word play in a sentence. These two features are also widely applied in sentiment analysis and other NLP (natural language processing) fields. Qazvinian et al. [8] designed 3 groups of features based on content, network and twitter. In content based feature, both unigram and bigram features are involved. They converted each feature value to a log likelihood ratio and evaluated the performance of system on individual group of feature set and the combined feature set. Kleinberg et al. [20] developed an online fake news detection system. They only focused on the linguistic features of the content including N-grams, punctuation, psycholinguistic, readability and syntax features. The performances of their SVM model over each type of feature and combined features are evaluated and compared. Readability features achieved the best performance, followed by the mixed feature

sets.

In this work, I propose to use features including content, propagation and user based features. In content features, a novel strategy is to use POS (part of speech) to represent the role of each word in a sentence. Also, Twitter specific features such as hashtags and URLs are utilized to train the model. For detailed information, please refer to section 3.

2.4 Machine learning models

Most machine learning models used in rumour detection studies are supervised classification algorithms. Accuracy is the common criteria to evaluate the performance of ML models. Whereas precision, recall and F1 score are also important indicators when judging rumour detection system. Because the distribution of rumour and non-rumour are more likely to be unbalanced, it is more scientific to analyse the proportion of False Positive and False Negative.

Ma et al [18] applied RNN (Recurrent Neural Networks) to implement the rumour identification in social networks. This machine learning technique is capable of exploring hidden characteristics of input data in absent of labeled features. In the evaluation step, they extracted two datasets. Both of them are from mainstream social platform namely Twitter and Sina Weibo. They used Snopes for Twitter and rumour busting function of Sina Weibo to get the data labeled. Their system achieved an accuracy of 88% on Twitter information and 91% on Sina Weibo information. Kwon et al. [19] explored the propagation pattern of rumour on Twitter. They investigated three features of rumour: temporal, structural and linguistic. These features are extracted by using time series fitting model and network structure. They compared the accuracy of three classification ML models: Decision Tree, Random Forest and SVM. The results suggest that Random Forest achieves the best accuracy of up to 90%. Zubiaga et al. [7] compared the performance of SVM, Random Forest, Naïve Bayes, Maximum Entropy and CRF on rumour detection system based on Twitter dataset. The result indicates that CRF achieved the best precision of 0.67 and F1 score of 0.61. Naïve Bayes achieved the best recall of 0.72 on their combined feature set. Ke et al. [21] implemented rumour detection on Sina Weibo with a hybrid SVM classifier along with random walk kernel and normal RBF kernel and achieved a remarkable accuracy of 0.913 which is higher than state-of-art approach. Their system holds 90% confidence to identify rumours after one day that rumour has burst out.

Although machine learning models can be tuned to detect specific rumours spreading in social media, it is difficult to use those models for future rumours as their properties change over time (e.g. source of rumour, rumour influence, keywords/hashtags, etc.). Therefore, we must also consider how those machine learning models perform when all types of rumours are considered in a generic pool of information. That is, we should focus on developing methods that can differentiate rumours from legitimate

information rather than specific results for predefined dataset. To investigate this, we look at not only the performance of false information detection for individual datasets, but also for the mixed dataset with different rumour topics in Twitter.

3. Methodology

The typical procedures for rumour detection system that applied in many previous works can be summarized to four main steps: data collection, feature extraction, machine learning model training and performance evaluation. Raw data should be preprocessed and then the features could be extracted. Features are the representation of raw data as well as the input to ML models. In my implementation, the data collection step is simplified in this work since the data retrieval and manually data annotation can be time consuming and laborious. A public available Twitter dataset is adopted as original data. It contains 5 individual rumour event datasets. Then the feature matrix of each dataset as well as the mixed dataset is extracted and split into training set and test set. Training set is fed into machine learning models for learning and delivering the testing set to the trained models should give the prediction of rumour or non-rumour for each tweet. Finally, several experiments are conducted to evaluate the performance over various ML models, different datasets, different training and testing set ratio, different features and new rumour events. The overall pipeline of this project is demonstrated in Figure 1. When testing with new rumours in practice, it follows the green arrows: after retrieved from API and feature extraction, directly sent to ML models for label prediction.

3.1 Dataset

PHEME, the Twitter rumour dataset used in this work is developed by Zubiaga et al. [24]. Unlike other works (Qazvinian et al. [8] and Castillo et al. [15]) defining an event as rumour and collecting data by entering associated key words to Twitter API, this dataset is constructed based on the information that has not been proved as rumour. Researchers first identify trending events to which substantial tweets are related. Secondly, journalists inspect the timeline of a trending event and identify the key tweets that deserve to analyse according to their number of retweets and other features that reflect a high potential of being widespread later. Then they verify and annotate the credibility of candidate tweets based on the predefined annotation scheme [25]. Finally, they also extract the conversations (i.e. replying tweets) associated with each original tweet for further analysis of rumour propagation. [24] Buntain et al. [16] also used PHEME dataset to detect fake news in popular Twitter thread.

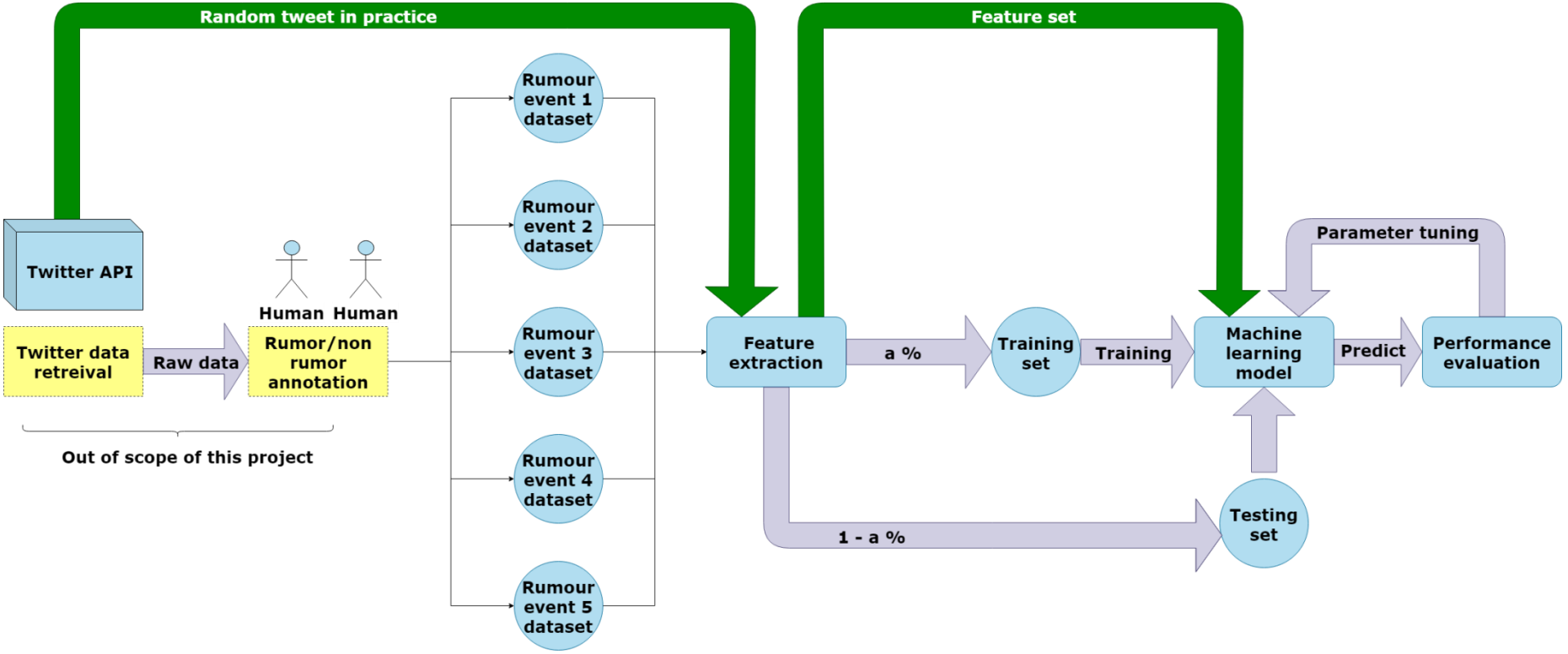

Figure 1: Pipeline for rumor detection

This dataset contains five rumour trending events. The statistics of them and the mixed dataset are shown in Figure 2.

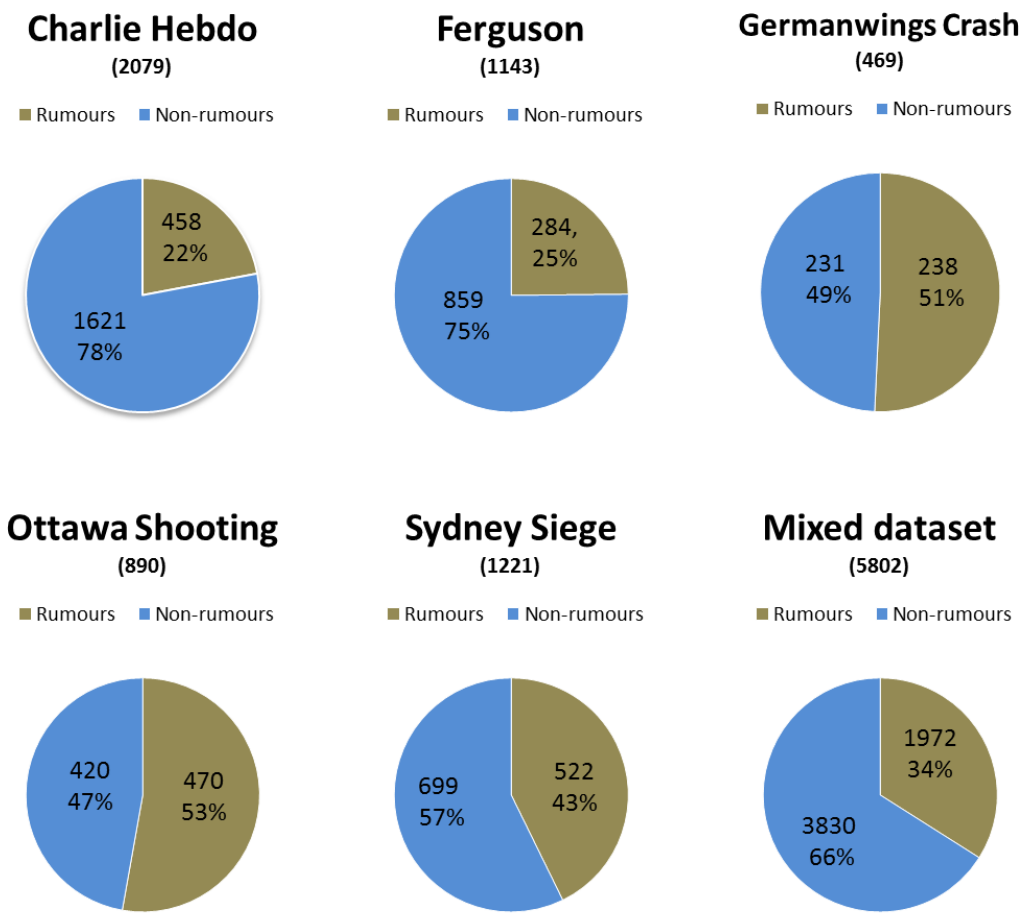

Figure 2: Distribution of rumours and non-rumours in individual and mixed dataset

Table 3 illustrates some examples of rumours and non-rumours of Ottawa shooting which happened on Ottawa's Parliament Hill in Canada, and one Canadian soldier died in this event. The first rumour example claims the event occurred at Canada's national war memorial which is contradicted with the fact that it happened on Parliament Hill. And the person got shot was a Canadian soldier instead of a ceremonial guard. The second rumour example asserts that there was a fire exchange between the shooter and hill security. This statement is apparently fabricated and intentionally created to mislead other users. By contrast, the non-rumour examples also discuss this event but do not conflict with the fact.

Table 3: Rumours and non-rumours examples from the text of Ottawa shooting dataset

Rumours	Label(1)	"Shots were fired at Canada's national war memorial in Ottawa and a ceremonial guard was shot, police say. http:\/\/t.co\/9KkQ1apxxW"
		"UPDATE | Shots were fired inside Parliament Hill and gun fire was exchanged with hill security. #cbcOTT #OTTnews"
Non-rumours	Label(0)	"If there is a bright spot in Ottawa today, it's that @Ottawaparamedic did an amazing job keeping that young Soldier alive."
		"You may want to stop and take a deep breath before tweeting insults at parliamentarians and reporters in #ottawa today, OK?"

3.2 Features

The features are the numerical representation of an example. A well-designed feature set is crucial for many machine learning models. In Twitter rumour detection field, a single example becomes a tweet object that contains all the information related to this tweet. A tweet object is a dictionary data structure including the content, user, entities etc. The goal of feature extraction is to extract useful features from each tweet object. I argue that there are several aspects deserving to investigate the credibility of information in social media as follows:

- The influence of the user who posts the tweet. Since most of the rumours are intent to propagate as far as possible in a user network, an influential account is an ideal carrier for rumours. This factor can be measured by the follower/following count, user whether verified etc.
- The certainty degree of the information. Many rumours contain strong modal words such as “must”, “should”. This linguistic characteristic can be leveraged to distinguish rumours from non-rumours which usually contain weak modal words such as “may”, “might”. Apart from modal words, there are many other words can reflect the certainty degree. As long as the appearances of those words are detected, the overall certainty degree can be estimated.
- The reaction of other users. The reactions include the popularity of the information and the opinion of users who comment this information. Popularity can be measured by re-tweet count, favorite count etc. Opinion can be measured by sentiment analysis towards the replying tweets etc.

In order to select the best features that reflect the characteristic of bad rumours, I applied most of the 15 features proposed by Castillo et al. [15] They analysed 68 features based on message, user, topic and propagation and assessed the effectiveness of each feature. Finally, they proved there are 15 best features that can achieve high performance. The defect of their feature set is the missing of more advanced linguistic features which are widely investigated in NLP related literatures [7,

8, 20, 26]. Therefore, word vector and POS tagging vector are added into my feature set to capture the characteristic of rumour content more accurately. The 14 features used in this work are listed in Appendix A. Most of them have been applied in other studies whereas a new feature “favorite count” is rarely analysed in previous work. The reason of involving feature is that it could indicate the popularity level of a tweet which can be used to differentiate the rumours and non-rumours. And it works similar as retweet count and reply count indicating the propagation of a tweet. The dimension and data type of each feature can be found in Appendix B. The dimension of U-BOW feature varies when the size of rumour event dataset changes. And Boolean type of data is represented as 0 or 1 in feature set.

- **Tweet cleaning and tokenization**

Before extracting U-BOW and POS feature, the text of tweet must be cleaned and tokenized. Stock market tickers, retweet text "RT", URLs, hashtag sign “#”, punctuation, happy and sad emoticons are removed from the text of tweet. Then I applied NLTK [28], a text processing library based on Python, to tokenize and stem the cleaned tweets. Rather than using bigram or higher order N-gram (for example, “I really like reading books”, after bigram tokenization becomes “I really”, “really like”, “like reading”, “reading books”), this work applies unigram tokenization (the previous example becomes “I”, “really”, “like”, “reading”, “books”). And words like “trouble”, “troubling”, “troubled”, all become “troubl” after word stemming. Table 4 shows examples of tweet cleaning and tokenization.

Table 4: Examples of tweet cleaning and tokenization

Original tweet	Cleaned tweet
"UPDATE | Shots were fired inside Parliament Hill and gun fire was exchanged with hill security. #cbcOTT #OTTnews"	'updat', 'shot', 'fire', 'insid', 'parliament', 'hill', 'gun', 'fire', 'exchang', 'hill', 'secur', 'cbcott', 'ottnew'
"BREAKING NEWS: ONE SOLDIER HAS DIED AFTER OTTAWA SHOOTING, SUSPECT(S) STILL AT LARGE - http:\/\/t.co\/lXFiYaVImZ "	'break', 'news', 'one', 'soldier', 'die', 'ottawa', 'shoot', 'suspect', 'still', 'larg'

- **Unigram Bag of Words (U-BOW)**

This feature represents the appearance of each word in a tweet. Previous works in NLP field also use this feature to distinguish rumours from non-rumours. Hamidian et al. [26] applied WEKA’s String To Word Vector along with N-gram tokenizer and generated unigram tokenized word form and bigram tokenized word form. Zubiaga et al. [7] also used Word2Vec [29] to create a 300 dimensional word vector as the content feature. Qazvinian et al. [8] analysed unigram and bigram lexical patterns towards the text of tweet when they develop feature set for rumour detection system. They also proved that the precision and recall of system relied on content feature outperform the system relied on network feature and Twitter specific feature. Therefore lexical pattern is an effective indicator of rumours. P´erez-Rosas et al. [20]

extracted unigrams and bigrams and applied bag of words to represent the words and segments in news article when they develop fake news detection system. They got 651 and 1378 dimensional N-gram features in their two fake news datasets, respectively. And both of them achieved competitive performance compared with other features.

In this work, I develop the unigram bag of words to represent the appearance of each token in a tweet, a similar approach as in [30]. We consider the whole dataset to extract this feature of each tweet. For example, in Ottawa shooting this event dataset, we have 890 tweets in total. Now, we assume it only contains two tweets: “Two people were killed. Three people injured” and “Five people were shot. Shooter disappeared.” Table 7 explains the bag of word representation of this dataset.

Table 5: Example of bag of words

Original tweet	Cleaned tweet	Vocabulary list	BoW vector
"Two people were killed. Three people were injured"	['two', 'peopl', 'kill', 'three', 'peopl', 'injur']	[two', 'peopl', 'kill', 'three', 'injur', 'sent', 'hospit']	[1, 2, 1, 1, 1, 0, 0]
"The injured had been sent to a hospital."	['injur', 'sent', 'hospit']		[0, 0, 0, 0, 1, 1, 1]

As shown in Table 5, a vocabulary list is created and it contains all the unique tokens in the whole dataset. The length of our word vector equals to the length of vocabulary list. Each element in vector matches the corresponding token in vocabulary list, and the value stands for the number of occurrences in this tweet. Since this example only contains two tweets, we only got 7-dimensional Bag-of-word (BoW) vectors. If the dataset contain more tweets, the length of word vector will increase dramatically. Table 6 lists the dimensions of this feature across different datasets.

- **Part of Speech Tagging (POS)**

Part of speech tags stand for the grammatical categories. A token in a sentence can be parsed by its definition and its role in a sentence. For example, “book” can be categorised as a “noun” part of speech tag. Such POS tagging can extract the semantics of the content grammatically. Many previous studies also embedded this method for text parsing in NLP field. Hamidian et al. [26] used CMU Twitter POS tagger to parse the tweet text and presented 0 as unseen and 1 as observed. Bontcheva et al. [31] developed a complete pipeline called TwitIE (Twitter Information Extraction) customised to Twitter for information extraction. The input text data will go through language Identification, TwitIE Tokeniser, Normaliser, Stanford POS Tagger, and Named Entity Recogniser. They developed a remarkable system that is able to handle Twitter specific messages which are usually short, noisy with high-frequency occurrence of emoticons, abbreviations, URLs and hashtags. They applied an adapted Stanford tagger for POS feature extraction with extra tagger labels equipped including retweets, URLs, hashtags and user mentions. Zubiaga et al. [7] also utilised

TwitIE to extract POS taggers. Since TwitIE is based on Java rather than Python, I failed to adopt their method. Instead, I utilised NLTK [28] again to implement POS tagging and an example is shown in Table 7.

Table 6: Dimensions of U-BOW feature in different datasets

	Charlie Hebdo	Ferguson	German Crash	Ottawa Shooting	Sydney Siege	Mixed dataset
Number of tweets	2079	1143	469	890	1221	5802
Dimensions of U-BOW	2976	2309	961	1516	1945	5585

Table 7: POS tagging using NLTK

Original text	Parsed text using POS tagging
"Witness tells CNN gunman shot one of two soldiers standing guard at war memorial in Ottawa."	[('Witness', 'NNP'), ('tells', 'VBZ'), ('CNN', 'NNP'), ('gunman', 'NN'), ('shot', 'VBD'), ('one', 'CD'), ('of', 'IN'), ('two', 'CD'), ('soldiers', 'NNS'), ('standing', 'VBG'), ('guard', 'NN'), ('at', 'IN'), ('war', 'NN'), ('memorial', 'NN'), ('in', 'IN'), ('Ottawa', 'NNP'), ('.', '.')]

Each token is labeled with a POS tag. For example, the first token “Witness” is tagged with “NNP” standing for noun, proper and singular. Second token “tells” is tagged with “VBZ” standing for verb, present tense and 3rd person singular. The full POS taggers definitions are available at [35]. In my implementation, I extract POS feature based on each rumour event dataset. First, extract all the tokens in a dataset and create vocabulary. Second, implement POS tagging of unique vocabulary and create a tags vocabulary. Finally, apply the bag of words again towards the tag of each token to represent the tags distribution of a tweet.

3.3 ML models

- **Models in Scikit-learn**

Most of existing works apply supervised machine learning model to solve classification problem such as online rumour detection, opinion mining and sentiment analysis. Various machine learning models have been proven to be able to achieve satisfactory accuracy including support vector machine (SVM), random forest, decision tree and logistic regression etc. Inspired by previous works [7, 18, 19, 21], I select six machine leaning models under scikit-learn library [32] which are logistic regression (LR), decision tree (DT), k nearest neighbour (KNN), linear discriminant analysis (LDA), Gaussian Naïve Bayes (GNB) and support vector machine (SVM). After preprocessing from last stage, the feature set is available. Once the proportion of training set and test set is specified, the system could randomly separate the dataset according to desired split ratio. Next step is normalizing the training set feature which is also called feature scaling. Scikit-learn’s StandardScaler [32] is applied to fit the data. The formula is shown in Equation (1).

$$k = \frac{x - u}{s} \quad (1)$$

where x is a sample of feature vector over all tweets in training set, u is the mean of x, s is the standard deviation of x and k is scaled feature values. Therefore, all the values in training set are centered around 0 and have unit variance. Next, fit the test set data using the formula above but with the u and s derived from training set. Then, feed the training set data to each ML model and use test set to evaluate the performance of each individual model. Finally, the performance summaries of all ML models are reported automatically (for detailed information, please refer to section 4).

- **Gaussian Naive Bayes**

Beside ML models in scikit-learn which are like black box models, a Gaussian Naive Bayes model is constructed for this project from scratch. Naive Bayes is widely applied for text classification problems such as spam email classification. The logic behind Naïve Bayes is briefly shown as Equation (2).

$$P(y|x_1, x_2 \dots x_n) = \frac{P(x_1, x_2 \dots x_n|y)P(y)}{P(x_1, x_2 \dots x_n)} \quad (2)$$

The formula above is Bayes theorem. $x_1, x_2 \dots x_n$ are the attributes which are also the feature values of a tweet. *n* equals to the number of features. *y* is the prediction(0 – non-rumour, 1 – rumour). The maximum probability of *y* given observation $x_1, x_2 \dots x_n$ is desired. A key hypothesis is all the features are independent with each other. Therefore, the formula can be transformed as Equation (3).

$$P(y|x_1, x_2 \dots x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1, x_2 \dots x_n)} \quad (3)$$

For a training set, $P(x_1, x_2 \dots x_n)$ remain the same no matter we calculate $P(y = 1|x_1, x_2 \dots x_n)$ or $P(y = 0|x_1, x_2 \dots x_n)$. So this term can be ignored and only need to compare the numerator $P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)$. An issue in practical implementation is that when *n* is large, and $P(x_n|y)$ is supposed in the range [0, 1], the multiplication of a number of probabilities have the value approaching 0. This could result in floating point underflow (Python cannot present such small number and will regard it as 0). To address this issue, I apply Log Probabilities:

$$\log(P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)) = \log P(x_1|y) + \log P(x_2|y) \dots \log P(x_n|y) + \log P(y)$$

Both $\log X$ and X reach the maximum at the optimal X, and the multiplication of probabilities is converted to the summation of log probabilities to prevent the value close to 0. Therefore, the ultimate objective is to find *y* that maximize $\log P(x_1|y) + \log P(x_2|y) \dots \log P(x_n|y) + \log P(y)$ as predicted class.

To calculate $P(x_1|y)$, the tweets that belong to a single class should be focused upon. Since feature value can be any number in a wide range, it is impossible to count the occurrence frequency of a value to derive the probability. Therefore, it is assumed that the values in feature vectors follow a Gaussian distribution. As long as mean and standard deviation of feature values are calculated from training set, the Gaussian probability density function can be derived. Thus, given any feature value in the test set, the probability of it can be estimated by Gaussian distribution density function shown as Equation (4).

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \quad (4)$$

where μ and σ are the mean and standard deviation a feature vector from training set, respectively. x_i is a value of corresponding feature in test set. i represents the index of feature.

To calculate $P(y)$, only need to measure how many “1”s and “0”s in class vector and get the proportion of each class as probabilities.

The pseudo-code of Gaussian Naive Bayes is shown in Algorithm 1:

Algorithm: Gaussian Naive Bayes (GNB)
Inputs: Training dataset TR, Testing dataset TE
Output: A class vector C representing the prediction of each example in testing set
Procedures:

1. Read the training dataset TR
2. Calculate the probabilities of class $P(y = 0)$ and $P(y = 1)$
3. **For** each class in TR:
4. **For** each feature vector:
5. Calculate mean μ and standard deviation σ
6. Establish Gaussian density formula G
7. Read the testing dataset TE
8. **For** each tweet in TE:
9. **For** each feature value :
10. Input to G to get $P(x_i|y)$
11. **For** each class:
12. Calculate $\mathrm{P} = \log P(x_1|y) + \log P(x_2|y) \ldots \log P(x_n|y) + \log P(y)$
13. Predict this tweet’s class that maximize P
14. Output prediction vector C

Algorithm 1: GNB

4. Experiment

To investigate the effectiveness and accuracy of our proposed pipeline, features and ML models, five sets of experiments are carried out to evaluate the performance over different machine leaning models, different datasets, different training/testing set separation ratios, different feature sets and testing new rumour events using different features. To eliminate the undesired fluctuation on performance caused by randomly selected training sets. All the experiments are executed 5 times. And the results are derived by averaging the 5 iterations. And the default dataset for experiments based on Charlie Hebdo which contains 2079 tweets except for second experiment assessing performance on different dataset. The default size of training set is set to 85% unless the proportion is required to change in experiment 3. The default feature set is the combined feature set. There are four critical performance indicators widely applied in numerous machine learning related literatures, namely accuracy, precision, recall and F1 score. Higher values of these four indicators imply a better performance. Precision measures the proportion of rumours correctly predicted among the tweets we predict as rumours. Recall measures the proportion of rumours correctly predicted among the tweets that are actual rumours. However, there is a tradeoff between precision and recall and that is why F1 score is introduced to get the weighted average of precision and recall. Another two indicators utilized in this work are FN and FP rate indicating the error rate among all the examples. The definition of these indicators are summarised in Table 8.

Table 8: Definition of accuracy, precision, recall, f1 score, FN and FP rate

	Predicted Rumour(1)	**Predicted Non-rumour(0)**
Actual Rumour(1)	True Positive (TP)	False Negative (FN)
Actual Non-rumour(0)	False Positive (FP)	True Negative (TN)
Accuracy	$\frac{NO.tweets\ (predicted = actual)}{NO.all\ tweets}$	
Precision	$\frac{TP}{TP + FP}$	
Recall	$\frac{TP}{TP + FN}$	
F1 score	$\frac{2 \times precision \times recall}{precision + recall}$	
FN rate	$\frac{FN}{NO.all\ tweets}$	
FP rate	$\frac{FP}{NO.all\ tweets}$	

4.1 Performance of various ML models

This experiment is conducted to using various machine learning models on the same dataset under same settings. The goal is to compare the performance of ML models from three aspects, namely precision / recall / f1 score, accuracy on training / testing sets and run time.

Figure 3 shows the performance of three groups of ML models：1. Gaussian Naïve Bayes constructed in this project. 2. Six ML models under scikit-learn library. 3. Five ML models applied by Zubiaga et al. [7] The GNB constructed by me achieve a high precision but the recall and f1 score are much lower than most of models. ML models in scikit-learn outperform those in other groups and indicate a significant improvement over previous work [7]. And the logistic regression under scikit-learn achieves highest performance with 0.82 precision, 0.85 recall and 0.83 f1 score.

Figure 4 illustrates the accuracy of my GNB model on test set and the accuracies of six models on training and testing set. Most the models achieve excellent results on testing set with accuracies over 0.8. And the accuracy on training set of ML models under scikit-learn also reasonably higher than that on testing set. The logistic regression results in the highest accuracy of 0.88 on testing set.

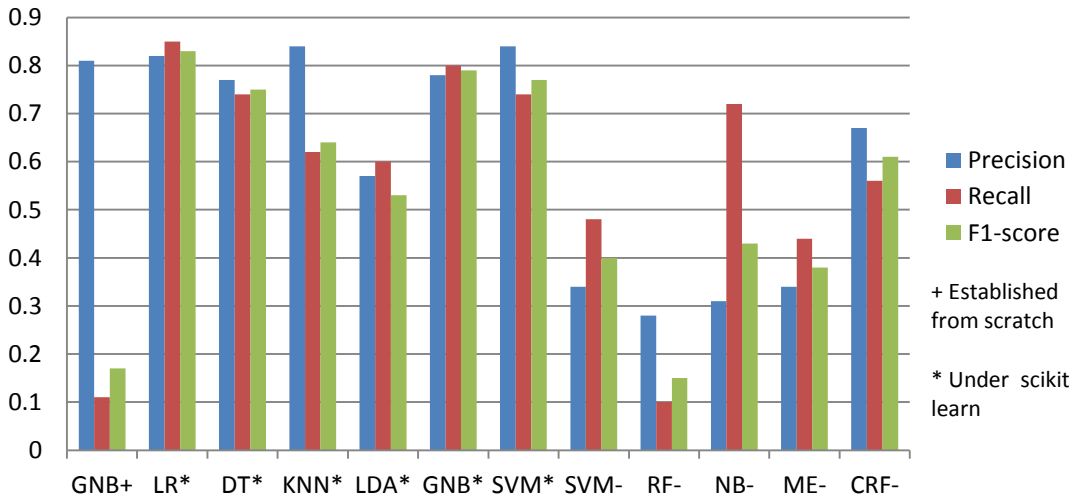

Figure 3: Precision, recall and f1 score of ML models

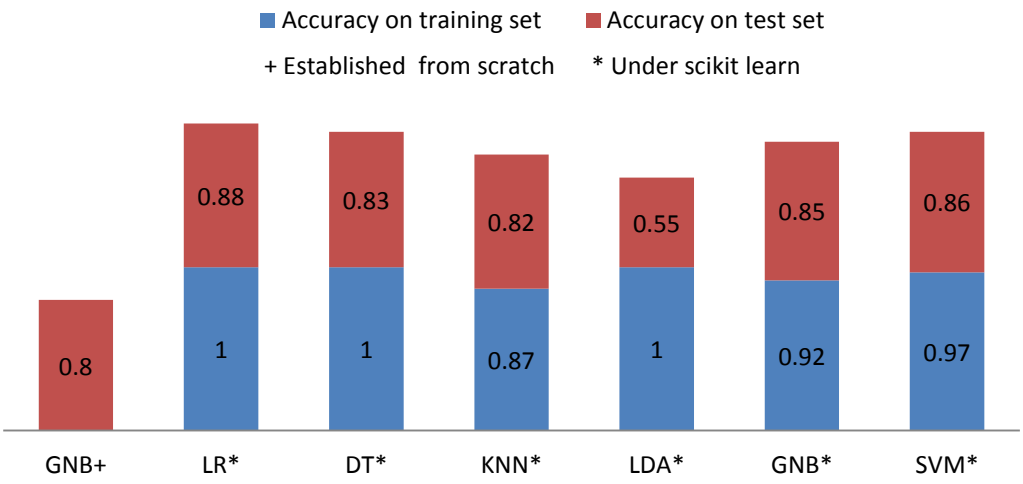

Figure 4: Accuracy performance on training and testing set of various ML models

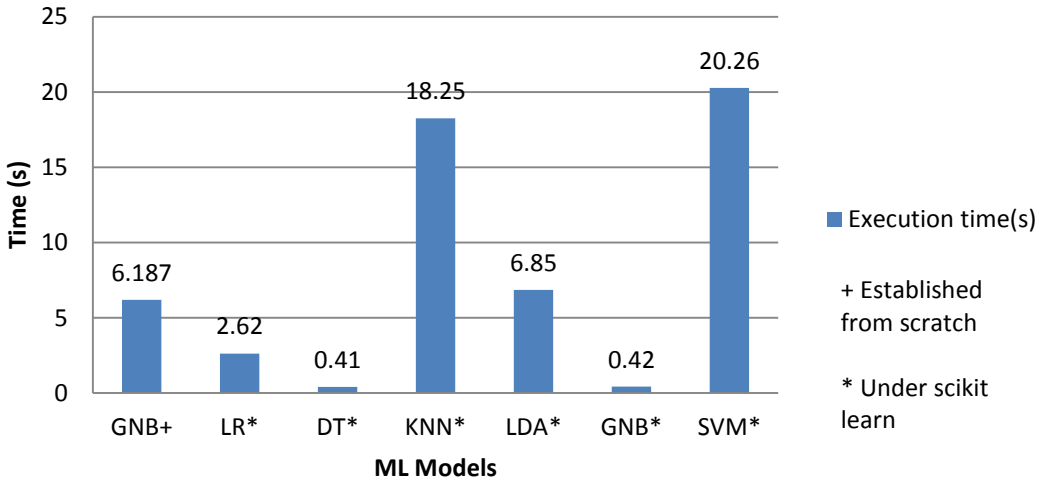

Figure 5: Execution time of various ML models

Figure 5 shows the execution time of each ML models under scikit-learn and my GNB model. Due to the inefficiency of code and the extension to log probabilities as mentioned in section 3, GNB model built in this project is beaten by GNB in scikit-learn. From the figure, the Decision Tree and Gaussian Naïve Bayes cost much less computing power than other models with a run time of 0.41s and 0.42s, respectively. So the most computationally efficient models are DT and GNB under Scikit-learn.

Figure 6 shows the confusion matrices of six ML models under scikit-learn, which has been plotted using the *seaborn* library in Python. The horizontal axis stands for the predicted label and vertical axis represents the actual label.

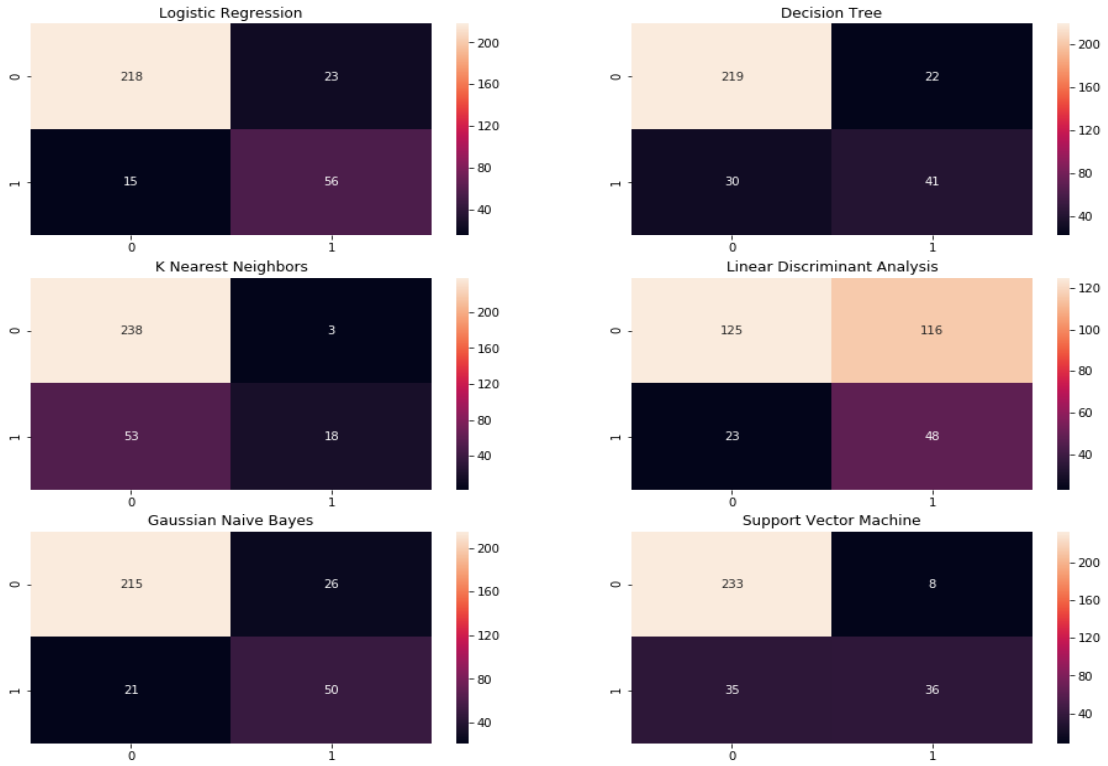

Figure 6: Confusion Matrices of ML models

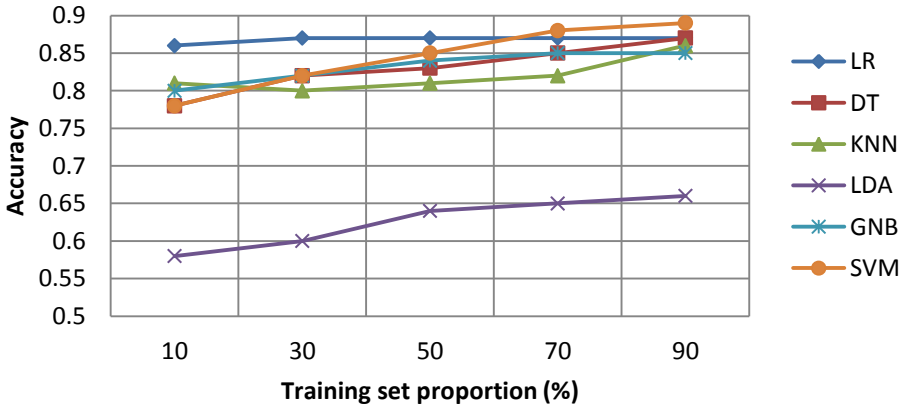

Figure 7: Accuracy under different training/testing set ratio

4.2 Impact of training/ testing set split ratio

In most of previous studies, a reasonable training set size is always between 70% and 90%. In order to assessing the impact of training set size, this experiment to designed to evaluate the performance with stepwise increased training set proportion. The results is shown in Figure 7.

As the proportion of training set rises from 10%, the accuracies of all the ML models increase gradually. After roughly 80% training set proportion, the curve flatten out which yields that 80% or higher is the ideal range of training set size to get optimal

accuracy for most of learning models.

4.3 Performance on various datasets

Since the propagation characteristic and the content features can be distinct between different rumour event datasets. The dimension of feature sets will also change because of the unigram bag of words feature relied on the unique vocabulary of whole dataset as mentioned in section 3. The variation of feature set may have some impact on performance. To assess the adaptive features of ML models with respect to the change of datasets, this experiment is conducted to get the performance over five individual dataset in PHEME and the mixed dataset of each ML model. Figures (8 - 13) shows the results for each dataset.

It can be found that most of ML models in Scikit-learn achieve similar performance when the rumour events vary. There is no sudden drop of performance at a rumour event. Thus, all the models are adapted to the change of rumour event dataset.

4.4 Performance on a feature level

This experiment is designed to assess the performance of various ML classifiers using different feature sets. In this work, there are three distinguished feature types namely, user-based and propagation-based features. This experiment bases on each individual feature set and the three types combined feature set. The performance is shown in Table 9.

From the table, it can be noticed that, in general, content based and combined feature sets achieve better performance than user and propagation based features. Combined feature set is roughly as good as content based feature set with a slight improvement. LR achieves the majority of highest accuracy and f1 score in various feature sets. Therefore, LR is the optimal model that well adapted to the change of feature sets.

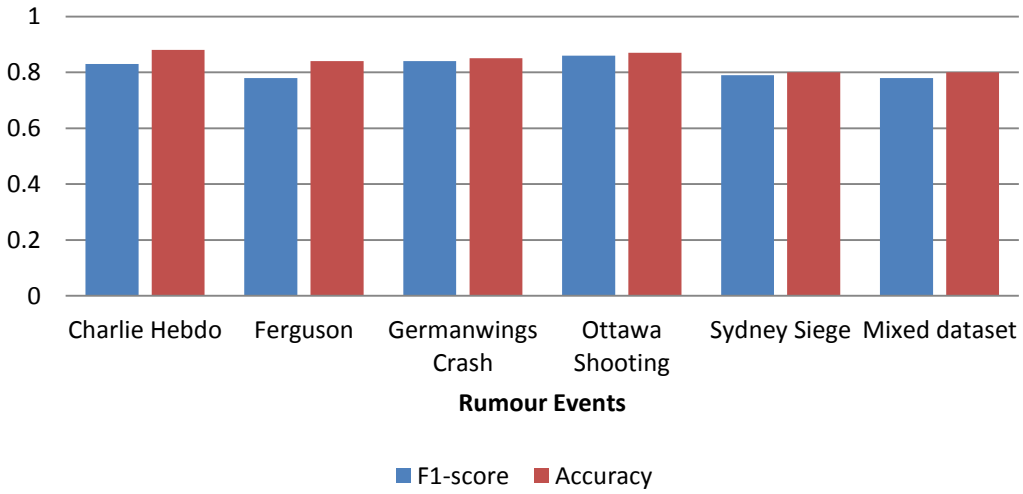

Figure 8: Performance of LR on different dataset

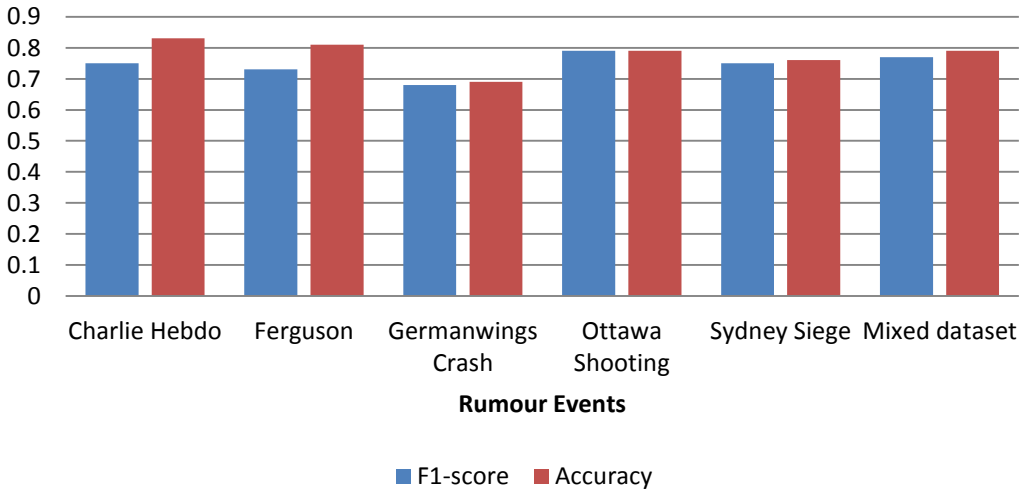

Figure 9: Performance of DT on different dataset

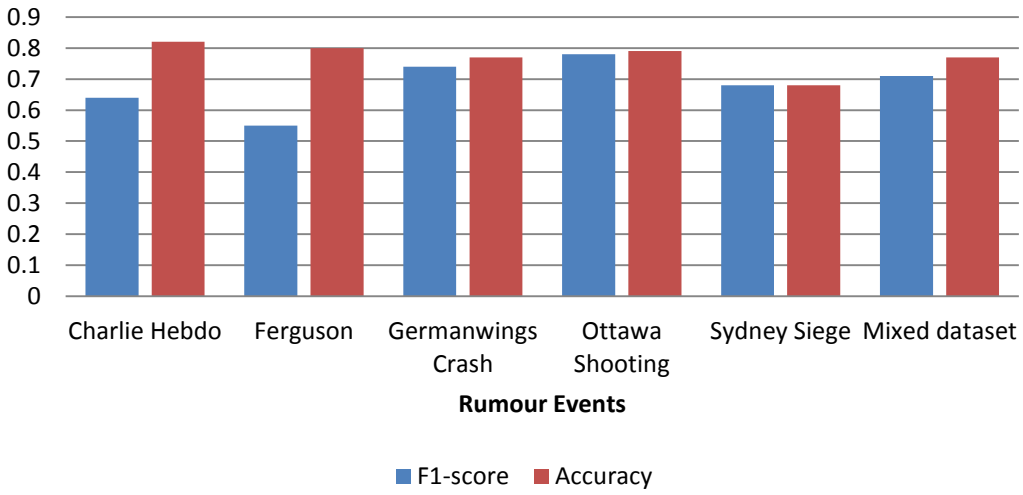

Figure 10: Performance of KNN on different dataset

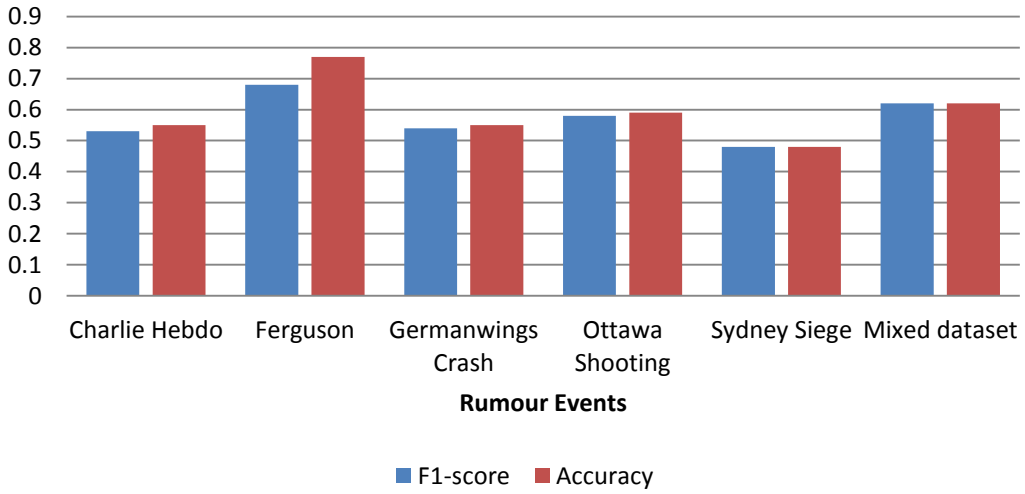

Figure 11: Performance of LDA on different dataset

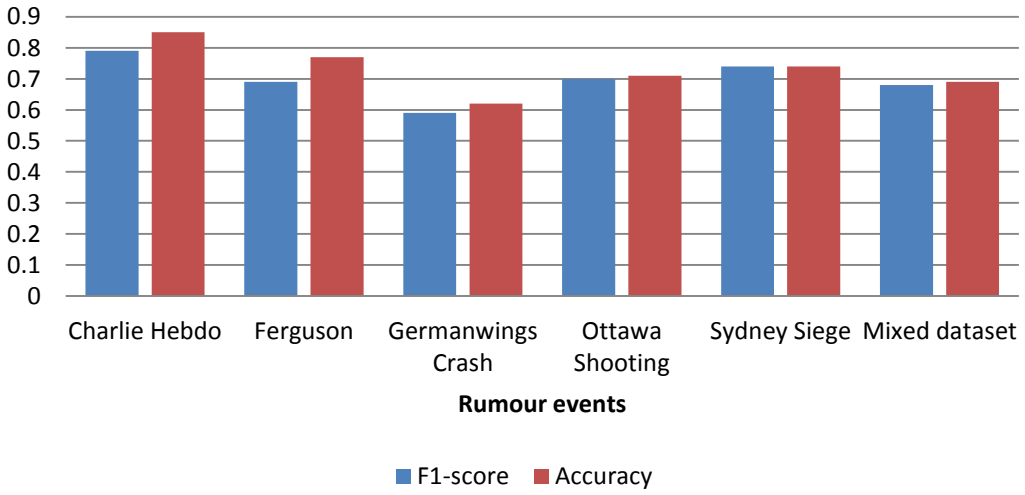

Figure 12: Performance of GNB on different dataset

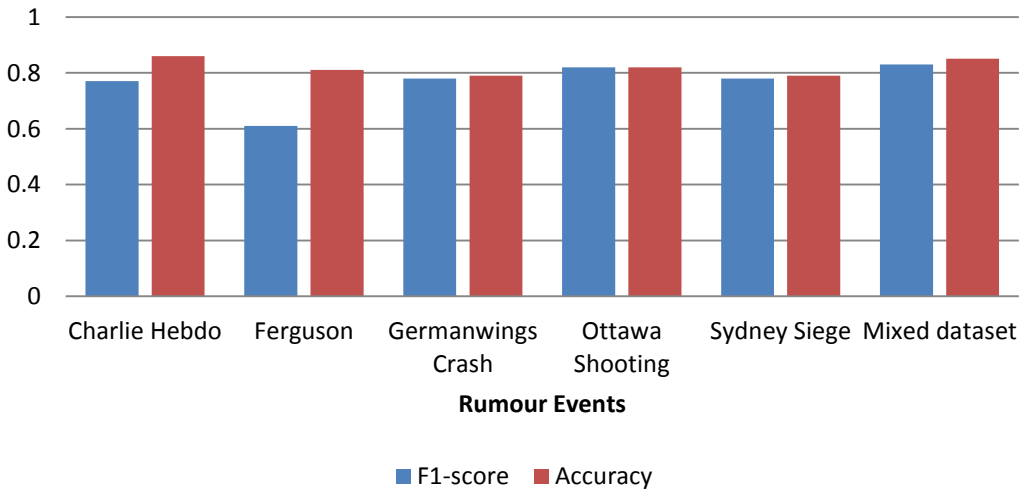

Figure 13: Performance of SVM on different dataset

Table 9: Performance on different group of feature set

Classifier	User		Propagation		Content		Combined	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
LR	0.77	0.68	**0.78**	0.68	**0.87**	**0.85**	**0.88**	**0.85**
DT	0.72	0.73	0.74	**0.74**	0.85	0.82	0.85	0.81
KNN	**0.79**	**0.78**	0.77	0.75	0.81	0.77	0.82	0.78
LDA	0.77	0.68	0.77	0.68	0.54	0.57	0.55	0.59
GNB	0.27	0.17	0.25	0.15	0.85	0.82	0.85	**0.85**
SVM	0.78	0.69	0.77	0.67	0.86	**0.85**	0.87	**0.85**

4.5 Testing new rumour events using different features

Differing from experiment 4 that testing 15% tweets within the same rumour event, this experiment uses 3 existing rumour events (Charlie Hebdo, Ferguson, Germanwings Crash) to train the model and uses 2 new rumour events (Ottawa Shooting, Sydney Siege) to test the model. Only by conducting this way can simulate the real task of rumour detection system in practice. The FN and FP rate are indicated in the Table 10. Please note, blue bold text is the lowest value and red bold text is the highest value for the given classifier.

FN% stands for the proportion of predicted non rumour but actual rumour. FP% stands for the proportion of predicted rumour but actual non rumour. Lower FN and FP rates are desired result. However, in practice, we may prioritise to accept low FN% rate without considering the FP%. Because if a non-rumour was classified as a rumour, it is less damaging to the entities involved, whereas with high FN%, actual rumours will bypass detection and spread false information. Therefore, we should focus on lowering the FN% as a priority over lowering the FP%. As shown in the table (i.e. red and blue bolded texts), it can be found that when trying to detect new rumour topics, the content based features achieved the lowest FN% for most of ML models and GNB achieves a remarkable FN% of 0.02 using the propagation based features.

Table 10: FN and PN rate on different group of feature set

Classifier	User		Propagation		Content		Combined	
	FN%	FP%	FN%	FP%	FN%	FP%	FN%	FP%
LR	**0.47**	0.001	0.47	0.002	**0.28**	0.11	0.29	0.11
DT	**0.38**	0.11	0.32	0.1	0.28	0.07	**0.28**	0.07
KNN	0.39	0.06	**0.36**	0.06	**0.44**	0.03	0.44	0.03
LDA	0.46	0.38	**0.47**	0.0005	**0.26**	0.21	0.32	0.16
GNB	0.36	0.07	**0.02**	0.47	0.39	0.1	**0.39**	0.1
SVM	0.45	0.01	**0.47**	0	**0.43**	0.004	0.43	0.004

The accuracy of testing known rumours and unknown rumours is compared which is shown in Figure 14. The first case (coloured in blue) uses all five events combined from the PHEME dataset with 85% training set size (i.e. detecting the remaining 15% unlabeled tweets). The second case (coloured in red) uses three events (a total of 3691 tweets) for training and two events (a total of 2111 tweets) for testing. The second case tests detection accuracy for ML models trained using known runours to detect new rumours with different topics. The results show that is we are detecting known rumours, then it is much easier to identify the content of the tweets and detect them. On the other hand, the content-based features are not very useful for new rumour topics, so the detection accuracy significantly reduced. In summary, it is more accurate to collect some new rumours for the training purpose and then detect them, rather than using ML models trained with other rumour topics. However, this can still be a challenge collecting the rumour information without having the fact check available to identify them.

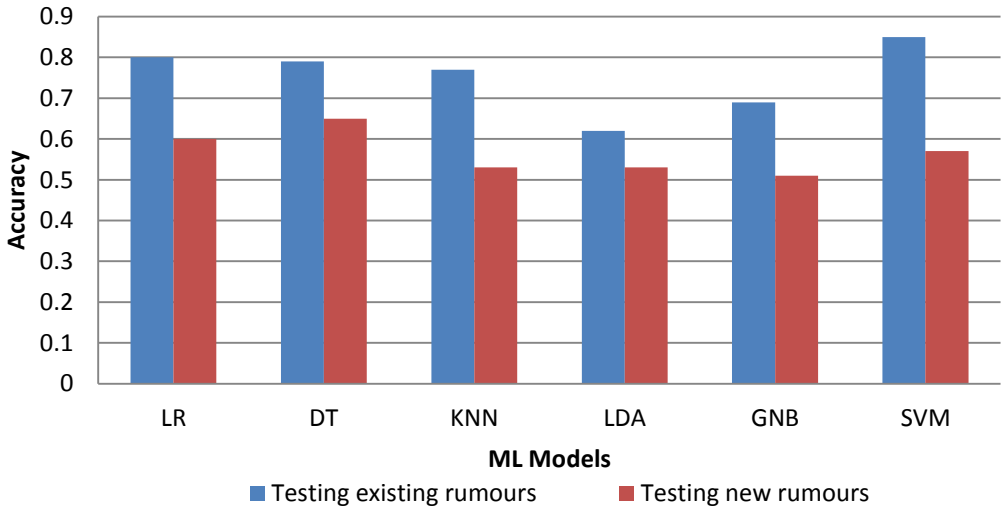

Figure 14: Accuracy comparison testing existing vs. new rumours

5. Discussion

A key hypothesis of this project is the veracity of all the tweets can be verified based on solid sources. There are some circumstances that during the early stage after an event takes place, no evidence is available to verify the authenticity of specific posts at that moment. The rumour detection system proposed in this work does not make predictions on such dataset. Instead, it can only work for dataset that the each tweet has already been labeled as rumour or non-rumour.

The implementation of this project follows the typical pipeline of twitter rumour detection. The variables in each step may be different with other works. For example, other researchers might use different datasets, different feature sets and ML models. How to select the most useful feature and the most effective model for a specific rumour event is a challenging problem and needs a lot of experiments to verify. In the future, I argue there are several aspects worth investigating as follows:

- **Sentiment Analysis.** Some NLP technique can be employed to extract the sentiment information of a tweet such as Positive, Neutral, Negative or we can use an averaged sentiment score. Hamidian et al. [26] applied the Stanford Sentiment system [33] to extract this content based feature.
- **Opinion Mining.** A further step of this work is to investigate the opinion of a tweet towards a rumour such as accept it, skeptical, deny it or neutral rather than just rumour or non rumour. This way the problem changes from binary classification to multi-class classification. The ML models should be modified to adapt to the change. For example, implement one vs. all instead of one vs. one in logistic regression and use Softmax technique to represent the probabilities of an example belonging to each class. Qazvinian et al. [8] defined a tweet having three types of opinions towards a tweet: confirm, deny and doubtful. Hamidian et al. [26] categorised tweet's opinion into four classes: endorse, deny, question and neutral.
- **Reactions.** Since the PHEME dataset used in this work also contains all the comment tweets of each original tweet but they are ignored in this work. So a future direction is taking advantage of them to create another feature regarding the reaction of original tweet such as average sentiment score of reactions etc.
- **Deep Learning.** With massive data created every day, traditional ML models can reach the bottleneck easily in terms of performance. Deep learning algorithms (such as deep neural network or recurrent neural network) should be effectively applied to boost the performance to a higher level when the number of examples (i.e. number of tweets) is enormous.

6. Conclusion

In this project, an automatic ML-based Twitter rumour detection system is designed using a public available dataset PHEME containing 5802 tweets across five rumour events. 14 features across user-based, propagation-based and content-based are designed as the input of ML models. 6 ML models under Scikit-learn and a GNB model built by myself are leveraged for rumour classification. From the five experiments, we observe the overall best performance is given by logistic regression (LR) model which achieves an accuracy of 0.88, f1 score of 0.83 and a competitive execution time of 2.62s. The ML models under Scikit-learn indicate a significant improvement over previous work. [7] The optimal training set size turns out to be between 80% and 90%. All the models under Scikit-learn remain robust when the dataset changes. Content-based features are more effective than user and propagation based features. When testing with new rumour events which are common cases in practice, GNB under Scikit-learn yields the lowest FN rate which is consistent to requirement that most of the rumours should be detected. However, directly feeding the new rumours into the model trained by other rumour events will not give high accuracy. The best practice is to train the model with some new rumours and then predict the label. Only by this way, the model is able to adapt to the characteristic of a specific rumour event and achieve a relatively good accuracy.

Reference

[1] Molina, B. (2017) “Twitter overcounted active users since 2014, shares surge on profit hopes”, USA Today, [online] Available: https://www.usatoday.com/story/tech/news/2017/10/26/twitter-overcounted-active-users-since-2014-shares-surge/801968001/ [Accessed: 30 March 2019]

[2] Isaac, M., Ember, S. (2016). "For Election Day Influence, Twitter Ruled Social Media". The New York Times. [online] Available: https://www.nytimes.com/2016/11/09/technology/for-election-day-chatter-twitter-ruled-social-media.html [Accessed: 30 March 2019]

[3] Kumar, S., West, R., & Leskovec, J. (2016) “Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes.” *Proceedings of the 25th International Conference on World Wide Web*, 2016.

[4] Mendoza, M., Poblete, b., & Castillo, C. (2010) “Twitter under crisis: Can we trust what we rt?” *1st Workshop on Social Media Analytics (SOMA ’10).* ACM Press.

[5] Zhao, Z., Resnick, P., & Mei, Q. (2015). “Enquiring minds: Early detection of rumours in social media from enquiry posts.” *Proceedings of the 24th International Conference on World Wide Web*, pages 1395-1405. ACM.

[6] Webb, H., Burnap, P., Procter, R., Rana, O., Stahl, B., Williams, M., Housley, W., Edwards, A., & Jirotka, M. (2016). “Digital wildfires: Propagation, verification, regulation, and responsible innovation”. *ACM Transactions on Information Systems*, 34(3).

[7] Zubiaga, A., Liakata, M., & Procter, R. (2016). “Learning reporting dynamics during breaking news for rumour detection in social media.” *arXiv preprint arXiv:1610.07363*.

[8] Qazvinian, V., Rosengren, E., Radev, D. R., & Mei, Q. (2011). “Rumour has it: Identifying misinformation in microblogs”. *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1589-1599). Association for Computational Linguistics.

[9] DiFonzo, N., & Bordia, P. (2007). “Rumour, gossip and urban legends” *Diogenes*, *54*(1), 19-35.

[10] Zannettou, S., Sirivianos, M., Blackburn, J., & Kourtellis, N. (2018). “The Web of False Information: Rumors, Fake News, Hoaxes, Clickbait, and Various Other Shenanigans”. *CoRR*, abs/1804.03461.

[11] Kumar, S., & Shah, N. (2018). “False Information on Web and Social Media: A Survey”. *CoRR*, abs/1804.08559.

[12] Horne, B. D., & Adali, S. (2017). “This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news”. *11th International AAAI Conference on Web and Social Media*.

[13] Bessi, A & Ferrara, E. (2016) “Social bots distort the 2016 us presidential election online discussion”. *First Monday， Volume 21, Number 11 - 7 November 2016*

[14] Subrahmanian, V. S., Azaria, A., Durst, S., Kagan, V., Galstyan, A., Lerman, K., ... & Menczer, F. (2016). “The DARPA Twitter bot challenge”. *Computer*, *49*(6), 38-46.

[15] Castillo, C., Mendoza, M., & Poblete, B. (2011). “Information credibility on twitter.” *Proceedings of the 20th international conference on World wide web* (pp. 675-684). ACM.
[16] Buntain, C., & Golbeck, J. (2017). “Automatically identifying fake news in popular twitter threads”. *2017 IEEE International Conference on Smart Cloud (SmartCloud)* (pp. 208-215). IEEE.
[17] Yang, F., Liu, Y., Yu, X., & Yang, M. (2012). “Automatic detection of rumour on Sina Weibo”. *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics* (p. 13). ACM.
[18] Jing M., Wei G., Prasenjit M., Sejeong K., Bernard J J., Kam-Fai W., & Meeyoung C.,(2016) “Detecting rumours from microblogs with recurrent neural networks”. *IJCAI*.
[19] Kwon, S., Cha, M., Jung, K., Chen, W., & Wang, Y. (2013). “Prominent features of rumor propagation in online social media.” *2013 IEEE 13th International Conference on Data Mining* (pp. 1103-1108). IEEE
[20] Pérez-Rosas, V., Kleinberg, B., Lefevre, A., & Mihalcea, R. (2017). “Automatic detection of fake news”. *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 3391--3401)
[21] Wu, K., Yang, S., & Zhu, K. Q. (2015). “False rumors detection on sina weibo by propagation structures”. *2015 IEEE 31st international conference on data engineering* (pp.651-662).IEEE.
[22] Zubiaga, A., Liakata, M., Procter, R., Hoi, G. W. S., & Tolmie, P. (2016). “Analysing how people orient to and spread rumours in social media by looking at conversational threads”. *PloS one, 11*(3), e0150989.
[23] Silverman, C. (2015). “Lies, damn lies, and viral content: How news websites spread (and Debunk) online rumors, unverified claims and misinformation”. Tow Center for Digital Journalism, 168(4), 134-140.
[24] Zubiaga, A., Liakata, M., Procter, R., Hoi, G. W. S., & Tolmie, P. (2016). “Analysing how people orient to and spread rumours in social media by looking at conversational threads”. *PloS one*, *11*(3), e0150989.
[25] Zubiaga A, Liakata M, Procter R, Bontcheva K, Tolmie P (2015). “Crowdsourcing the Annotation of Rumourous Conversations in Social Media”. *Proceedings of the World Wide Web Conference Companion. Web Science Track*; 2015.
[26] Hamidian, S., & Diab, M. T. (2015). “Rumor detection and classification for twitter data”. *Proceedings of the Fifth International Conference on Social Media Technologies, Communication, and Informatics (SOTICS)* (pp. 71-77).
[27] Ghenai, A. & Mejova, Y. (2017) "Catching Zika Fever: Application of Crowdsourcing and Machine Learning for Tracking Health Misinformation on Twitter," *2017 IEEE International Conference on Healthcare Informatics (ICHI) 10.1109/ICHI.2017.58* (pp. 518-518).
[28] Bird, S., Loper, E. & Klein, E. (2009), “Natural Language Processing with Python.” *O’Reilly Media Inc.*
[29] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). “Distributed representations of words and phrases and their compositionality.” *Advances in neural information processing systems*, pp. (3111-3119).

[30] Harrington, P. (2012). *Machine learning in action* (Vol. 5). Greenwich: Manning. pp. (61-82)
[31] Bontcheva, K., Derczynski, L., Funk, A., Greenwood, M., Maynard, D., & Aswani, N. (2013). “Twitie: An open-source information extraction pipeline for microblog text.” *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013* (pp. 83-90).
[32] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). “Scikit-learn: Machine learning in Python”. *Journal of machine learning research*, 12(Oct), 2825-2830.
[33] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). “Recursive deep models for semantic compositionality over a sentiment Treebank”. *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1631-1642).
[34] Sun, H. (2019) “twitter rumour detection.” *GitHub,Repository.* [online] Available: https://github.com/Hao12123821/Twitter-Rumour-Detection- [Accessed: 10 May 2019]
[35] Chapagain, M (2018) “Python NLTK: Part-of-Speech (POS) Tagging [Natural Language Processing (NLP)]”. *Mukesh Chapagain Blog* [online] Available: http://blog.chapagain.com.np/python-nltk-part-of-speech-pos-tagging-natural-language-processing-nlp/ [Accessed: 16 May 2019]

Appendix

Appendix A – Feature list

Category	Features	Applied in other literatures	Description
USER	**IS VERIFIED**	[7,15,17,21]	Whether the user is verified by Twitter
	HAS DESCRIPTION	[15,17,21]	Whether the user has personal description
	FOLLOWERS COUNT	[15,17,21,27]	Total number of users who follow this user.
	FRIENDS COUNT	[15,17,21,27]	Total number of users that this user follows
	STATUSES COUNT	[7,15,17,21,27]	Total number of tweets the user has posted since this account was created
PROPAGATION	**TIME SPAN**	[7,15,17,21,27]	Time span between the posting time and the account registration time
	RETWEET COUNT	[17,27]	Number of tweets that re-tweet this tweet
	FAVORITE COUNT	-	Number of users who like this tweet
CONTENT	**HAS QUESTION MARK**	[7,15,27]	Whether the text contains question mark “?”
	HAS EXCLAMATION MARK	[7,15,27]	Whether the text contains exclamation mark”!”
	HAS URL	[8,15,17,21,26]	Whether the text contains URL
	HAS HASHTAG	[8,15,26,27]	Whether the text contains hash-tags ”#”
	UNIGRAM BOW VECTOR	[7,8,20,26]	Vector representation of the tweet content (number of occurrences of individual token)
	POS TAGGING	[7,8,20,26]	Vector representation of the grammatical categories of each token.

Appendix B – Feature dimensions and data type

Category	Features	Dimensions	Data type
USER	IS VERIFIED	1	Boolean
	HAS DESCRIPTION	1	Boolean
	FOLLOWERS COUNT	1	Integer
	FRIENDS COUNT	1	Integer
	STATUSES COUNT	1	Integer
PROPAGATION	TIME SPAN	1	Integer
	RETWEET COUNT	1	Integer
	FAVORITE COUNT	1	Integer
CONTENT	HAS QUESTION MARK	1	Boolean
	HAS EXCLAMATION MARK	1	Boolean
	HAS URL	1	Boolean
	HAS HASHTAG	1	Boolean
	UNIGRAM BOW VECTOR	961~5585*	Vector
	POS TAGGING	28	Vector

(Note: *marked dimension varies with the size of rumour event dataset)