#Social_Networks

V. SALMANOVICIUS	A. CHAN	J. JAIENTILAL	H. WANG	Q. FENG	J. GALLE
vs17622	ac14712	jj17584	hw17892	qf17615	jg17035
1759327	1427360	1756525	1758886	1758701	1764832

***Abstract*—The aim of this project is to explore and extract features that underly Twitter messages. The final product is a web application with a search engine to view informative visualisations, which are based upon a Twitter user. After requesting the Twitter timeline, the designed system pre-processes the user information, gives a LDA-based recommendation for similar users, performs a sentiment analysis of the tweets, and also presents a word cloud with words from the account's most recent tweets. The results are evaluated under a proposed framework. Additionally, LDA models trained with more tweets performed better and interesting statistical patterns were found in the data.**

***Index Terms*—Twitter, Topic Modelling, Sentiment Analysis**

I. INTRODUCTION

These days a vast majority of the people are not afraid to reveal themselves in the online world, sharing daily activities, opinions, political statements or any other personal information. Twitter alone generates a stream of more than 6000 tweets every second [1]. Year on year, the amount of people using social networks is still increasing in almost every country [2]. As a consequence, a huge amount of data is produced and can be captured carrying either disclosed or hidden correlated information.

Recent studies and their discoveries showed the vast amount of possibilities you can do with the captured data. To give an example, the heatmap of Strava which is an application that tracks and uploads the athletic activity of its users, gave away location of secret US army bases [3]. Another example is the well known leaked scandal in which Cambridge Analytica used the data of more than 50 million Facebook profiles without permission, in order to influence voter's opinion on behalf of politicians who had hired them [4].

The main question that arises from social networks is: what other stories publicly shared data can tell? Therefore, the aim of this project is to explore and extract features that underly Twitter messages. To achieve this, the following objectives are defined:

- Recommend user profiles based on their tweets.
- Perform a sentiment analysis of a specific user's tweets.
- Visualise a time-based evolution of topics based on tweets.

Following the specific goals defined above, the practical deliverable is a web application, that combines all of those. The web application can be accessed through (http://d32ggkx37ai485.cloudfront.net)

II. RELATED WORK

A. Topic modeling

Topic modeling is increasingly becoming popular for different areas, which LDA algorithm is being used as a standard tool to address such application [12]. Zhao and Jiang [13] studied topic modeling with Twitter data as a means of comparing against traditional news media. Using LDA, a collection of Twitter messages is modelled to identify topics, and a comparison using similarity measures is done against New York times articles. The findings are that both cover topics of similar categories and types, but the distribution over them are different. Lertsiwaporn and Senivongse (2017) [14] streamlined topics and associated sentiment in one visualisation, using the LDA model. The work was done in Twitter messages written in Thai language. The results demonstrate good performance on both, Topic modeling and sentiment analysis. This work was useful on a context where one needs a better view of public opinion on a particular keyword, or subject, during a period of time. This has inspired this project, to discover topics of a particular subject which are mentioned in tweets.

III. SYSTEM ARCHITECTURE AND DEPLOYMENT

Data science pipelines are useful only when (1) they are available on demand, in some cases, with (2) low latency. Furthermore, if the system is open to the public, (3) scalability, (4) cost and (5) security, might be of concern.

In order to address issues 1-5, we have deployed the product to the cloud, such that it is scalable, cost-effective and secure solution.

1) Architecture: The design of the architecture includes elements of serverless computing with auto-scaling handled by the cloud service provider.

The system consists of APIs exposed through Amazon API gateway, that handles load balancing, throttling, routing and message passing for us. An API consists of URLs parsed according to their use case. Each URL is handled as HTTP GET or POST request, invoking relevant $Lambda$ function that handles response, which could be a sentiment analysis request for a specific username or word cloud data request. Sometimes Lambda can handle the request without additional communication, but in most cases Twitter API is used with MongoDB queries. Typically, the response is a JSON object.

A notable component of the architecture is Amazon S3 Bucket that stores static files. These include site markup, styling and client code. Although we could have used $Lambda$

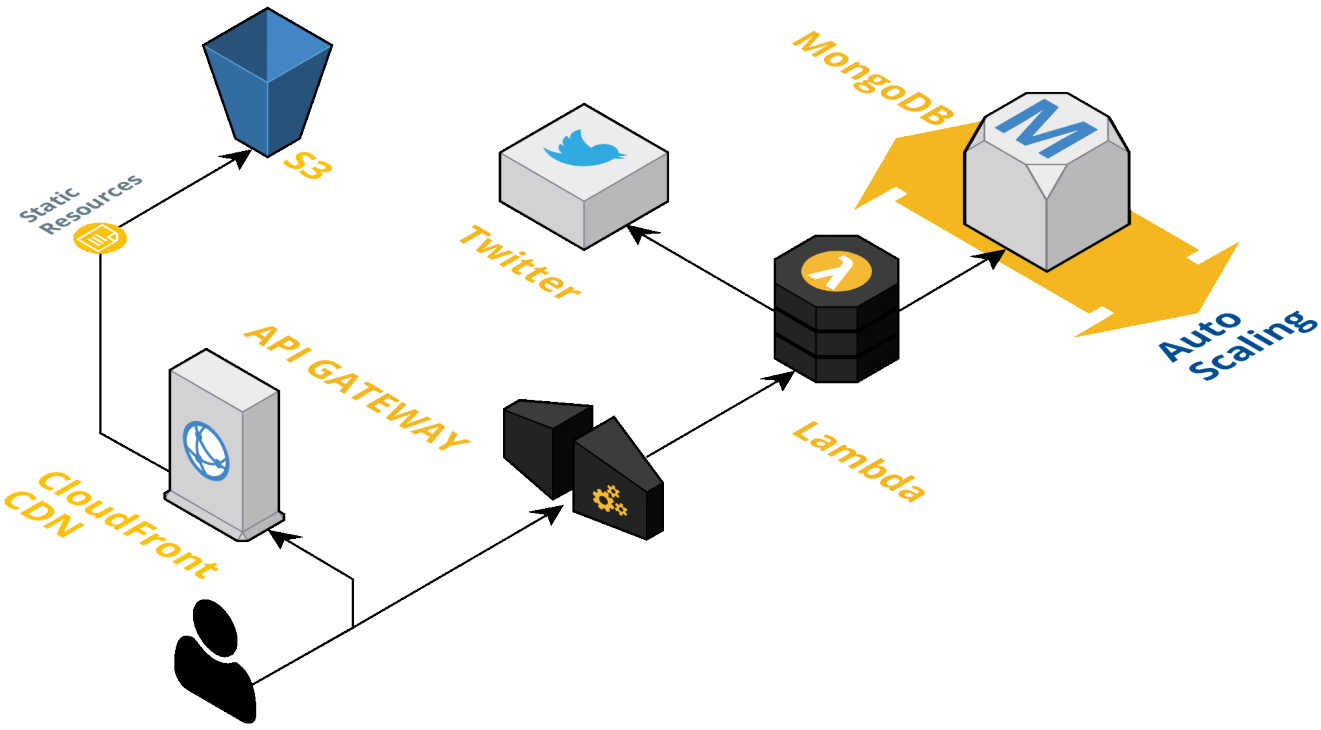

Fig. 1. Cloud Architecture.

to handle these as well, it is cost effective to handle such requests with minimal compute power.

The whole system architecture is visualised in figure 1

2) Scalability: We argue that the proposed architecture is fully scalable, except for Twitter API bottleneck, which is a dependency that can not be solved by Twitter.

In particular, S3 bucket is replicated across the globe for each availability zone. AWS lambda handles requests in parallel, with one request per function call, that can be deployed for each availability zone. MongoDB is autoscaled.

3) Security: In order to address security concerns, we made sure to place the database behind a firewall that whitelists only Lambda services. Lambda in turn, is not accessible from outside the protected network, except for communicating with the Twitter API. The API gateway serves as a layer between back-end and a client, but this is fully managed by Amazon, with our configuration for throttling and URLs.

Additionally, we configured the system components with strong passwords where possible. Also, the database was configured to perform weekly backups and Amazon to store logs with alerts on performance spikes above a certain threshold.

IV. Collection and Storage of Tweets

This section discusses about how data is gathered and stored into MongoDB. Figure 2 shows a pipeline of the data collection and storage process.

A. Twitter API

Twitter provides an open-access API, which allows users to collect tweets, user profiles, and related information. However, there is a request limitation for free API accounts. Users with a free account can only make a limited number of requests within a given time window [19]. Once the limit has been exceeded, the API will prevents users from making further requests until the time window refreshes. There are two type of APIs discussed below.

- **Twitter Stream API & REST API**: Twitter provides two types of APIs which are Twitter Stream API and REST API. They both allow users to collect twitter data by tracking the words or specific parameters. However, there is an essential difference between the two APIs. The stream goes forward and REST goes back in time. In other words, Stream API is suitable for doing analysis on the live data and REST API is better choice when users want to analyse the historical data.

All of the Twitter APIs respond with data encoded using JavaScript Object Notation (JSON).This contains information about the tweet, including author profile data, tweet message, hashtags, geotags, place, mentions, URL's and their titles. However, most of the attributes are often missing, since not all profiles include full details and tweets rarely contain additional information such as geotags or URL attachments [8].

We will need to collect three data sets: the stresming data, those users' timeline tweets and celebrities' tweet messages, because of the primary task of our project is to suggest users

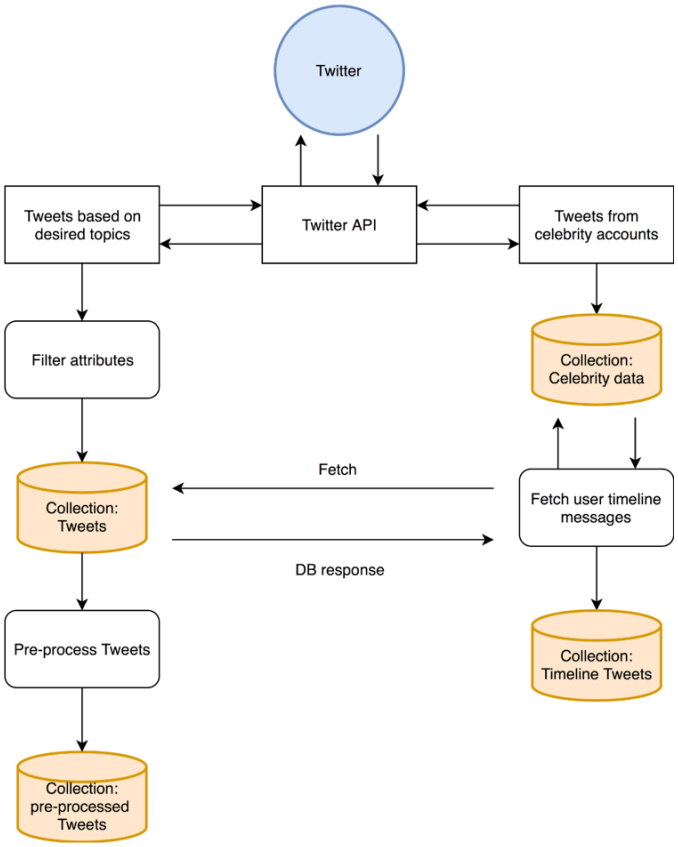

Fig. 2. Data collecting pipeline.

some accounts to follow based on similar interest. To gathering these data, both Twitter APIs are used in our project.

- **Collecting streaming tweets**: Twitter messages consist of 140 characters and contain millions of topics. The increasing size of the dataset makes it difficult for analysis. Therefore, in this case study, we will only focus on eight topics (football, university, Bristol, food, shopping, marvel, game and trump). According to the Twitter Help Centre, a hashtag is written with a # symbol at the beginning of a word. Hashtags are used to index keyword or topic on Twitter [10]. Moreover, Stream API provides a dictionary-based data collection function that the API can automatically filter the tweets by the keywords. Thus, we put eight topics with # symbol as a hashtag in the keyword list and let the API collecting the data based on it.

Fig. 3. an example tweet for topic 'Marvel'

- **Timeline tweets**: To identify the user's interest, it is neccessary to access their Twitter timeline. A user's timeline is an exhaustive collection of the person's 'tweeting' history. Twitter API provides a function to fetch a user's timeline. Using the Twitter API, 200 tweet messages for a small number of active users found in the streaming data collection and celebrities are collected.

By the end of this project, the total number of tweets we collected are shown in table I.

Type of tweets	Number of tweets
Live time tweets	64002
User's timeline tweets	78425
Celebrities' tweets	3000

TABLE I
TABLE OF COLLECTED TWEETS

B. MongoDB

MongoDB is a document-oriented Database Management System (DBMS) was chosen because of the following features:

- **Document Oriented**: stores information aggregated into collections, which subsequently is a group of documents. Each document is a JSON object.
- **Dynamic Schemas:** comparatively to SQL databases, they require to fix a schema before storing the data. Oppositely, NoSQL have a flexible schema, which the insertion of data can be executed without a predefined schema. Advantageuosly, development will be faster and code integration will be more reliable.
- **Scalability**: allows to scale horizontally by using a process denominated Key Sharding. This technique uses the key as the means of separating the data between the instances. Consequently, the reading and writing latency are decreased, because different requests, in the best case, will be processed by different instances.

The reasons to choose MongoDB for this project can be inffered from the aforementioned advantages. Firstly, Twitter API returns a JSON, so storing raw tweets can be done directly. Moreover, if the API changes (such as some extra fields in the returned JSON) then the structure is not needed to change, because this accepts natively a flexible schema. Secondly, horizontal scalability is provided, which is helpful for storing large amount of tweets, for fast query response.

V. Preprocessing

Preprocessing seeks to improve the quality of the representation of the data. This simplifies the analysis, hence is an essential procedure of the pipeline.

A. Lexer

Lexer is a component of compilers that translates source codes into a sequence of tokens. This concept is applied to converting raw tweets into a list of words.

Definitions of tokens: Although tweets can be simply splited into words at spaces, this method has several disadvantages compared with a lexer. 1) After the split at spaces, words need to be trimed to remove surrounding punctuations. An example would be hashtags followed by a colon. 2) Then other techniques such as regular expression must be used to

TABLE II
TOKEN DEFINITIONS

Token	Definition	Priority
Stop word	a|the|to|is|am|are|hello|...	1
Word	[a-zA-Z\-]+([']\[dst])?|([a-zA-Z]\.)+|[a-zA-Z]+\.	2
Number	[0-9\.\,]+	3
Hashtag	#[a-zA-Z_][a-zA-Z_0-9]*	4
Username	@[a-zA-Z0-9_]+	5
URL	(http://—https://)?[a-zA-Z0-9\./\+-\?\=%&]+	6
Space	[ ]+	7
Unknown	.	8

find out the type of words (hashtag, username, stop word etc.). 3) It does not perform well for tweets that are not properly segmented by spaces. For example, ellipses are frequently used to sperate two sentences without any spacees between the two sentences. A lexer can mitigate these issues without perplexing the implementation.

Definitions of tokens: Lexer can split tweets into predefined tokens (as shown in table II). The only modification to lexer for source code is an "unknown" token defined to be any character. This special token allows lexer to handle unexpected tokens in tweets, such as non-English characters and text emojis. Similar to lexers of compilers, this lexer also needs to solve ambiguities in token definitions. For example, the first token of tweet "Hello world!" can be a stop word "hello", an ordinary word "hello", or an unknown token "h". All the three outputs are legitimate according to token definitons. This ambiguity can be eliminated by 1) choosing token of the longest matching length 2) and choosing token with the highest priority in case of the smae matching length. Following these rules, a stop word token "hello" is choosen, because it has the longest matching length is 5 and priority of stop word token is higher than word token. After ambiguity is solved, regular expressions representing tokens can be translated into one deterministic finite automata which takes in characters of a tweet and generates tokens for it.

Refinement of tokens: Tokens recognized by lexer may need to be refined in a couple of ways. 1) For example, the last word in a sentence or quote is likely to contain a period or an apostrophe. A straightforward approach is to remove any punctuation at the end of a word token, but it may errorly remove dot at the end of an abbreviation or apostrophe following a owner name (students' for example). 2) This issue and a more general version of it can be mitigated by clustering words into groups that share the same context and have a small edit distance between each other. Context considered in this part is one token right before and after a word. Clustering can be done by a hashtable whose keys are contexts and values are lists of tokens that appear in certain context. To further group tokens in one list (or tokens sharing context), edit distance is used. After this clustering, different spellings of a word are likely to be unified to one. For instance, words "uk", "U.K.", and "U.K" are unified as "uk". In addition, plurals of some words are grouped with their single froms as well. This simple clustering refines some tokens produced by lexer, even though advanced models are needed to achieve better results.

B. Discover similar Tweets with Minhash

A large quantity of similar tweets are found in the data, such as reposted tweets and advertisements. Repeatitive tweets have considerable effects on the training results of models used in other sections, particularly when data set is not sufficiently large. It could be desirable to identify these tweets. MinHash is a method to efficiently estimate similarity of two sets, which could also be adapted to group redundant tweets.

Jaccard similarity: MinHash attempts to estimate the Jaccard similarities between many pairs of tweets. For two sets A and B, Jaccard similarity can be defined by

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

In order to apply it on bags of words (which may contain repeated words and thus are not necessarily sets), the above equation can be modified as

$$J(A,B) = \frac{2|A \cap B|}{|A \cup B| + |A \cap B|}$$

Despite modifications, the valued calculated by latter equation have similar meaning and the same range as previous one.

k-Shingles: A preparation step for MinHash is to convert sequences of tokens into sequences of k-Shingles. Each group of k consecutive tokens in tweets can be viewed as a k-shingle. The k selected for this project is 3. Each 3-shingle is further hashed into an 24-bit integer to represent it. The range of integer is picked to be less than the total number of unique 3-shingles found in data set. The 3-shingles may have distinct frequencies in data set which could undermine the performance of MinHash. But after hashing, integers represents 3-shingles tend to distribute more evenly and thus are more suitable inputs for MinHash than tokens produced by previous steps.

Difficulties in finding similar tweets: Treated as bag of integers marking 3-shingles, two nearly-duplicated tweets can be identified by their high Jaccard similarity. However, it is impractical to directly measure similarities for every pair of tweets, because the number of such pairs is huge.

MinHash: MinHash can address this issue by systematically sampling a few k-shingles from a bag. The way MinHash generates a digestion for each tweet is described as below. 1) First, a permutation of all possible k-shingles is defined. 2) Next, k-shingles in the permutation are enumerated. And their existence in a tweet are checked. This step stops until one k-shingle is found in the tweet. 3) Then, this procedure is repeated with different permutations to sample more k-shingle from the tweet. 4) Lastly, arrange these sampled k-shingles in one vector to form the tweet's MinHash signature $[kshingle_1, kshingle_2, ..., kshingle_6]$, where $kshingle_i$ is the k-shingle sample with ith permutation.

Implementation of MinHash: The second step of MinHash method involves enumerating all possible k-shingles, which can be inefficient. Given a tweet consisting of k-shingles, MinHash can be implemented as following. 1) Firstly, the bits of integer representing each 3-shingles in the tweet are

permutated. 2) Secondly, the smallest integer after permutation and its corresponding k-shingle are found. 3) Thirdly, the previous two steps are repeated. And 3-shingles found at step 2 are organized in a vector to form a MinHash signature. In this way, MinHash signature of a tweet can be calculated in $O(L)$ time complexity, where L is the length of a tweet.

Find nearly-duplicated tweets: A hashtable is used to enable tweet searching by MinHash signatures. Whenever a new tweet is added to the database, its minhash signature is calculated and searched in the hashtable to find previous tweets that are potentially duplicates of the new tweet. Since candidates for replication of new tweet are few, it is feasible to compute the Jaccard similarity between each candidate and the new tweet. With Jaccard similarity, whether new tweet is a duplication of previous tweet can be determined.

VI. Topic Modelling with LDA

This section will describe the model used to identify topics of tweets. The Project used the Latent Dirichlet Allocation (LDA) . LDA is a generative probabilistic model to discover topics from documents (i.e. tweets). The assumption of this model is that each document is a mixture of topics, and each topic represents a probability distribution of words. Hence, LDA can reveal the topic distribution for each tweet. Python Gensim library was used gensim [26] to train a LDA model. The process of the LDA is (see figure 4):

- **Creating a corpus**: before training the LDA model. The previous collection of tweets are tokenised and words are filtered.
- **Constructing the Topic Model with Gensim**: the first step involves creating a lookup table of word frequencies. Then, this is used to train the LDA model.
- **Identifying topics for each tweet**: the last step involves using our trained model to identify the main topic of the tweet. This is obtained by the most probable topic from the mixture.

VII. Sentiment Analysis

Key word based approach was used for sentiment analysis. We have use Valence Aware Dictionary and Sentiment Reasoner (VADER). This is a heuristical approach (lexical and rule-based). Which takes into account word weighting negation (words preceded by 'not'), use of ponctuation, emoticons and emojis. Every keyword is weighted to give a final sentiment. The model has built-in dictionary of weighted words [25].

VIII. Word Embeddings and Autoencoders

Direct text analysis is challenging, because of all the possible variations of the words, wheres numbers are unambiguous in their representation. For instance, 'go', 'going', 'goes' have the same semantic meaning, despite their different representations. In practice, this is encountered when comparing tweets.

There are several naive approaches one could take, such as comparing histograms of letter frequencies, or counting matching letters in two strings. However, such approaches are

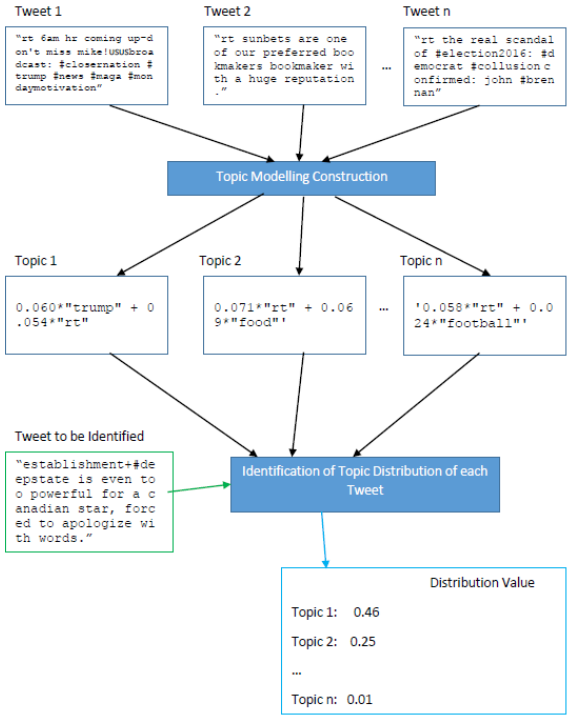

Fig. 4. LDA Process

semantic and context agnostic. A much better approach is to map the tweet to some space, such that the distance in that space corresponds to the similarity of tweets. This mapping is called an embedding and it requires building a model to perform translation of text to numbers [5].

One of the most popular open-source models to learn word embedding is Word2Vec by Mikolov et al. The model can be trained on large text datasets in an unsupervised manner to perform mapping from a single word to a vector [6]. The model is built on an assumption that two words in one sentence are more related than two random words in a document. The implication of this assumption is in that unrelated words that are close to each other, will be considered related, but on a large scale, this is not an issue, given a good quality of data.

In practice, the model is capable of finding interesting embedding given just a large set of text or tweets. We have experimented with the Word2Vec model pre-trained on 400 million tweets by Godin et al [7]. The pre-trained model was able to perform arithmetic queries such as $'King' - 'Man' + 'Woman' \approx 'Queen'$ or deciding that word $'cereal'$ does not fit into the sequence *'breakfast cereal dinner lunch'*.

The Word2Vec can only deal with single words only. Hence, it is not possible to perform inference of similarity measure between two tweets explicitly. In attempt to overcome this issue, we hypothesized that cumulative distance of similarities between each pair of words can be used as an implicit similarity between two tweets. However, the small test case has shown that the approach degrades quickly in terms of

performance and accuracy with the length of tweets. The time complexity of such approach is $O(n^2)$, while accuracy degrades simply because more noise arises with more inputs. This could be fixed by picking only the most strong or diverse keywords out of the tweets and calculating cumulative distance between them, but this seems to be a hard problem of its own. Since it requires a model to extract the most representative keyword of a tweet.

Although Word2Vec proved unsuccessful for sentences, there are other models, built on the same principles, for instance, Sentence2Vec or Doc2Vec. While the former can be used for tweet similarity, the latter can be applied to profiles directly. However, to the best of our knowledge, neither of the models are publicly available with pre-trained weights for social media texts.

We can see how such model can be useful to natural language processing tasks. However, embedding also allows for dimensionality reduction.

We know that each tweet is at most 280 characters, hence storing an arbitrary tweet requires a string of length 280 characters, this includes various languages and symbols. However, if we could encode a tweet as a vector of numbers that is smaller than the original string, we can achieve a new representation of the tweet and store it instead, decoding it back into original string on demand. A models that can perform this kind of dimensionality reduction are called autoencoders.

In many cases autoencoders are implemented with hourglass neural network architecture, that learns to represent input layer with significantly smaller hidden layer. After the training, the first half of neural network can be used as an encoder and the second half as decoder, where the state of smallest hidden layer is an embedding.

Such networks are optimised to reconstruct an input layer, hence the assumption that similar objects would have similar representation.

IX. Existing Twitter Dataset

In order to better understand the nature of Twitter data, we have decided to turn our attention to existing tweet datasets.

One of the few open source general tweet datasets is gathered by Yang and Leskovec [11]. The data consists of nearly 50 million tweets per month, collected in 2009 over the period of 6 months. The total dataset size is nearly 25 GB compressed. In order to work with in-memory tools, we have decided to use data only of June 2009.

Each data entry consists of three attributes: timestamp, username and tweet text. We have first used the timestamp to investigate temporal data features. The timestamps are visualised in figure 5. From the distribution of tweets per day, two anomalies are noticeable: (1) data for 15th of June is missing, and (2) the tweet distribution per month does not look natural, because the change of the distribution should be smooth. Instead an abrupt increase occurs at day 25 and 29. We hypothesize based on (1) and (2), that the data gathering system was offline for prolonged periods of time. Based on this intuition, the data collection issue was partially first resolved on 25th June and further improved on 29th.

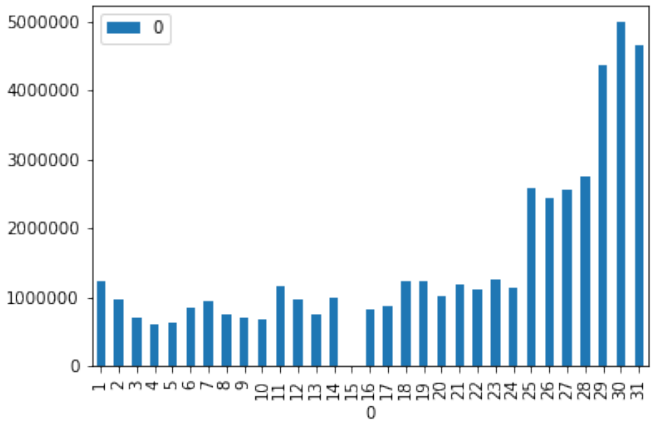

Fig. 5. Number of Tweets per day of June 2009 from the dataset [11]

Looking at the aforementioned dataset at an hourly scale, an increase in detail of temporal dynamics (see figure 5) can be observed. However, we have to bear in mind anomalies in figure 5. For this reason, normalisation by days is required. Furthermore, for a specific week day, the number of tweets can have abnormal variations, because daily routines are different.

The distribution of tweets over a 24 hour window, suggests that there are two peaks. Intuitively, the peaks could be lunch hours and midnight, however, this intuition is based on the assumption that tweets come from the same time zone. Another perspective is that this data can be modelled with a mixture of Gaussians, in turn this suggests there exists two data sources, where the peaks correlate to peak demand in different time zones, for example, US and Europe regions.

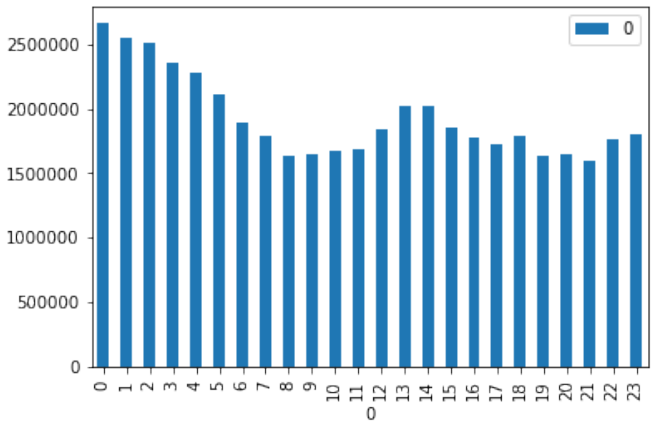

Fig. 6. Total number of tweets over 24 hours for June 2009 from the dataset [11]

The findings allows to make certain claims and decisions. For instance, more demand for the web application can be expected during midnight. This means, the system can be adjusted for efficiency during peak demand hours, making

the solution cost effective.

Analysing other data attributes, usernames were not particularly useful. Since any kind of information about users are constrained by Twitter API. Tweet texts on the other hand, are useful, but it is not immediately obvious on how they can be analysed to produce value.

One approach to analyze tweet texts, is to extract features from the text and evaluate the features directly. For instance, numbers found in the texts.

An interesting pattern can be found by looking at the leading digits of many numbers. Plotting the frequency of occurences of the first digit shows a Benford's law pattern (see figure 7). Benford's law states that given a distribution that spans vast order of magnitudes, the first digit is most likely to be a 1 [20]. Benford's law has been acknowledged by many publications [20]–[23].

Figure 7 shows the deviation from an ideal Benford's law distribution, in that Digit '2' is more frequent than '1'. The possible hypothesis for this nature, is that people, in modern language, use digit '2' as a short-version of the words 'to' and 'too'.

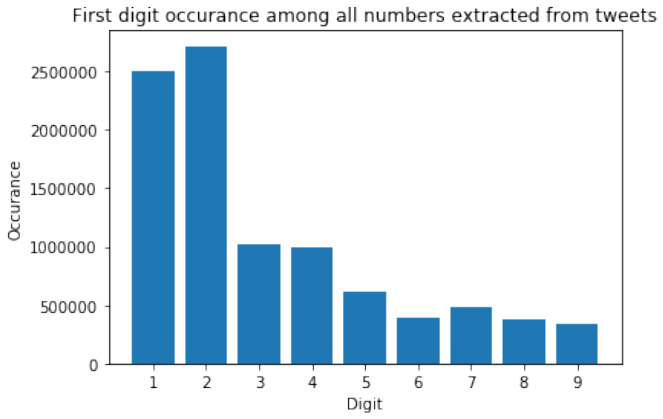

Fig. 7. Frequency of the leading digits from numbers extracted from the tweets.

X. Data Visualisation

This section discusses the visualisations presented, accompanied by their justifications. Figure 8 is a visualisation of a time based evolution of topics, inferred from the tweets. This shows the most tweeted topics over time. The steps towards this were: (1) identifying the topic of each tweet by using the LDA model; (2) counting the number of tweets of each topic for the different dates and times. The insights provided by this graph 8 are: the topic represented by the cyan colour, fills nearly half of the tweets; it is hypothesized that, the topics represented by the green and orange colour, lost their relevance over time, which correlates to the release date (30 April) of the Marvel movie Avengers Infinity Wars. Hence, it shows the decreasing relevance on the social network.

XI. Recommendation System

A. Simple Search Engine

A simple search engine is built to efficiently search tweets that contain certain keywords. Based on this search engine, a primative content-based recommendation system could be created.

Data structures: There are two data structures used in this search engine, both of which are based on *dicts* (implemented by balanced binary search trees).

1) Dictionary of keywords: A *dict* whose keys are keywords and values are inverted indice.

2) Inverted index: A *dicts* whose keys are ranks concatenated by tweet ids. The ranks decide the order in which these tweets will be displayed in search results and should be obtained by page ranking, but timestamps are used instead in this project.

Operations: These data structures can be created, updated, and searched efficiently.

1) Insert a new tweet (with its rank previously determined):

1a. For each keyword in the tweet, retrieve inverted index of the keyword using dictionary of keywords.

1b. And insert rank + tweet_id into the inverted index. It takes two requests to dicts for process of one tweet.

2) Search for a set of keywords: Search can be done by "merging" the inverted indice of keywords. Because the inverted indice is sorted and only overlap of these lists is output, search operation takes $O(min\{L_1, L_2, L_3, ...\})$ requests to *dicts*, where L_i is the length of inverted index of $keyword_i$.

3) Remove a tweet: For each keyword in the tweet, remove the tweet from the inverted index of the keywords. If the inverted index becomes empty, also remove the keyword from dictionary of keywords. This process takes $O(L)$ requests to *dicts*, where L is the maximum number of keywords in a tweet or maximum length of a tweet.

B. Simple Recommendation System based on Search Engine

A system can recommend tweets to users with the help of search engine and a naive Bayesian model.

Naive Bayes model: The naive Bayesian model is used in this project to guess keywords that users like. It assumes that every user is interested in a list of keywords. Given a keyword in the list, the user has a probability $P(keyword)$ to be interested in a tweet that contains this keyword. And the probability for each keyword in the list is independent. In addition, keywords outside this list do not affect users interests in any tweets. In the implement of this model, each user is associated with a short list of keywords with probabilities P and how many times they are measured N.

Recommendation: With list of keywords, recommendations can be made. 1) For each pair of keywords in the list, probability that it will be favored by user is estimated by

$$\prod_{i=1}^{2} P(keyword_i)$$

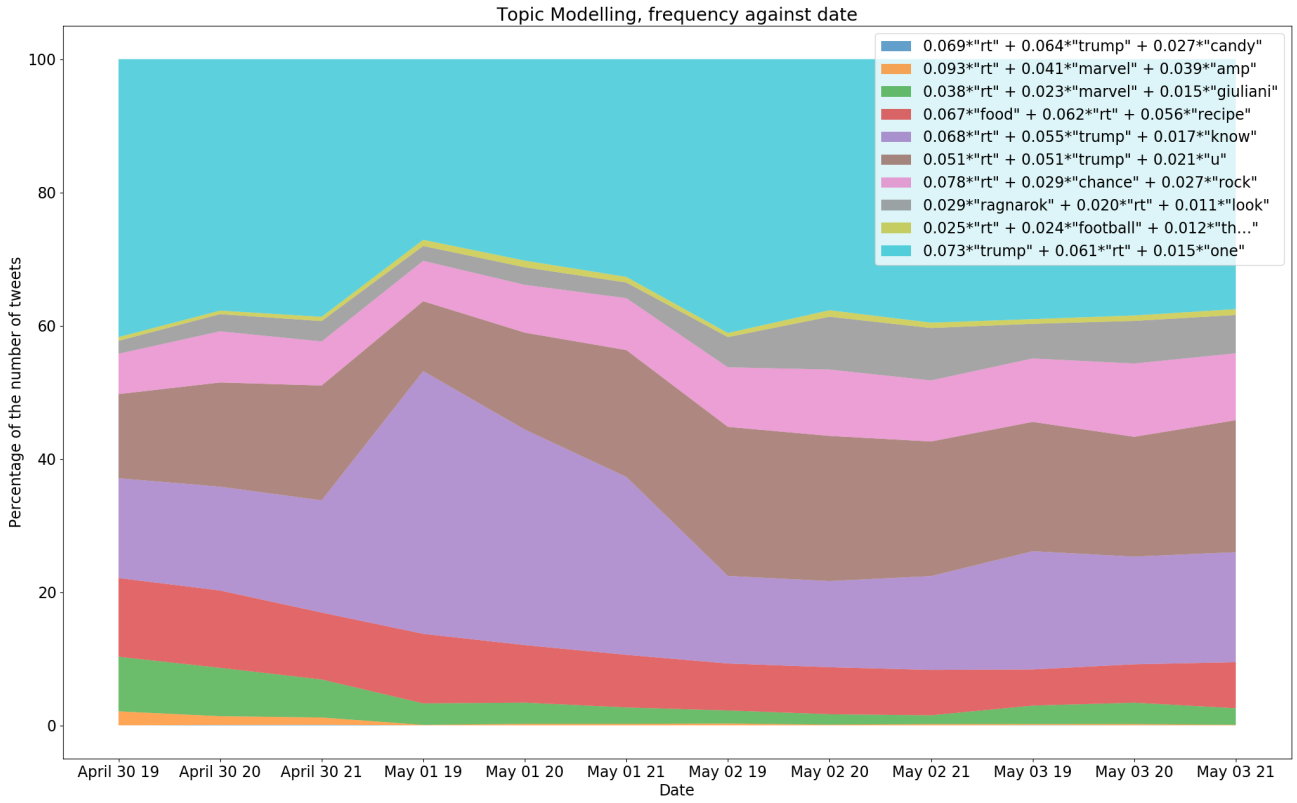

Fig. 8. Visualisation of the evolution of topics, where each topic is represented as a colour (normalised). A topic is a probability distribution of words. Date is formatted as Month-Day-Hour. Frequency is the total number of tweets.

2) Pairs of keywords are queried in the search engine until enough recommendations are found. Pairs with higher probability will be attempted before pairs with lower likelyhood. This procedure may trigger $O(N^2)$ requests to search engine, where N is the size of keyword list of the user. It means keyword lists have to be short.

Feedback: Keyword list can be updated in respond to feedback from user. Firstly, recommendations are displayed to let user choose the tweets that are interesting. Secondly, two statistics are gathered for each keyword in the recommendations. One is m, the number of tweets in which this keyword appears. The other is n, keyword frequency in tweets that are liked by the user. Thirdly, new keywords are added to the list. And probability P of each keyword in the list is updated by

$$P = \frac{P * N + n}{m + N}$$

In case of a full list, the keyword minimum P is removed from the list.

Disadvantages: This recommendation system have serveral disadvantages. 1) New users' keyword list are empty. So naive Bayesian model may not be able to make any suggestion. Although random tweets could be recommended for the user, it may take a long time to figure out user's tastes. 2) Performance of the recommendation system suffers from unrealistic assumptions made by naive Bayesian model and the lack of page ranking.

XII. Evaluation

This section will describe the attempted evaluation, to assess the performance of the models used. Due to inexistence of evaluation framework for topic modelling and sentiment analysis in social media context, we propose our own judgment with existing measures.

A. Topic Coherence of the LDA Model

Modelling human judgment is a difficult task, because two people can have different opinions regarding the importance of a topic [16].

Evaluating a topic model can be a non-trivial task, especially when working with unlabelled data, where the ground truth is unknown. However, the state of the art, exposes the topic coherence as a measure of a topic model. The score of a topic is based on the degree of semantic similarity between high scoring words in the topic. These measurements help distinguish between topics that are semantically interpretable and topics that are artifacts of statistical inference. However, in this case, different models will not be compared, but rather how different data set sizes used for training affect the LDA

model learning [15]. Being a topic V, that has a set of topic words, then the coherence of a topic can be calculated as:

$$coeherence(V) = \sum_{(v_i, v_j) \in V} score(v_i, v_j, \epsilon)$$

where the ϵ defines the smoothing factor, which was set to 1, the same value used by the original authors [15]. The score was calculated using the UMass measure, which is as follows:

$$score(v_i, v_j, \epsilon) = log \frac{D(v_i, v_j) + \epsilon}{D(v_j)}$$

where $D(x, y)$ calculates the number of documents that contain the words x and y, intuitively D(x) counts the number of documents that contain the word x. Noticeably, the $UMass$ measure uses the corpus, as it tries to measure how well the model has learnt from the data.

Hence, to evaluate the topic model, an arithmetic average of all topic coherence values is calculated. The results can be seen on Fig. 9. The average topic coherence increases as we train the LDA model with more tweets. Intuitively, the model learns better with more data, which makes the topic more interpretable.

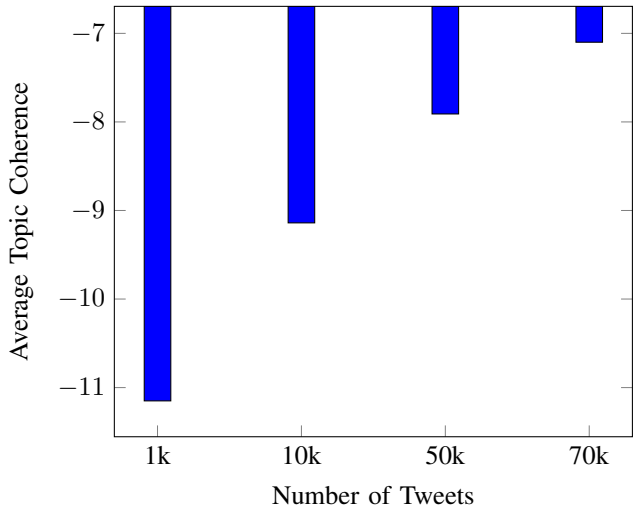

Fig. 9. Average topic coherence for different dataset sizes of Tweets with 10 Topics.

B. The difficulty of sentiment analysis

Sentiment analysis of short texts as Twitter message is challenging because of the limited contextual information that they usually contain. Moreover, Twitter messages are always writing in an informal way which include abbreviations, URL links, pictures and emoticons. Usually, we will delete these features during the preprocessing, since it is hard for a program to understand and judge the sentiment on it. However, removing the features mentioned is not ideal for sentiment analysis, these features are important for analysing emotion. This is evident in a tweet shown in figure 10 which is defined as neutral based on its text content, but if we consider the emoticons as well, it becomes a positive tweet.

Fig. 10. The tweet has different sentiment results with and without emoji

Furthermore, we found that current sentiment analysis did not capture the context of the tweet, for example, apologise can be considered positive or negative depending on the context. Another limitation of sentiment analysis is that it does not perform well for messages containing sarcasm, since sarcasm requires a degree of knowledge about the topic being discussed in a message. The difficulty of sarcasm also affects people, for instance, in dicusssion forums some people explicity say “not sarcasm” to avoid misunderstanding.

XIII. Improvements and Future Work

A. Other models

Besides LDA, both extensions for this model as well as other models have been developed. Hence, considering another model in order to improve the system might be worth examining. Another flexible framework for topic modelling is proposed by Dallas Card, Chenhao Tan and Noah A. Smith. [17]. This model is based on neural networks and it offers both a rapid exploration of documents as well as an easy way to incorporate prior information.

B. Collection and Storage of Tweets

The Twitter API poses some limitations, as mentioned before, not only on the architecture (as a bottleneck) but as a dataset size for our models. As shown in the evaluation of the topic coherence, its performance increases with more tweets. Hence, a future improvement would be by attempting to use the Twitter Premium API, which yields unlimited calls and therefore a larger collection of tweets. Furthermore, a comparison can be made regarding the performance of the models and check whether the topic coherence achieves an upper bound with larger dataset sizes, in which it does not improve much afterwards, and also studying how do a different number of topics affect the topic coherence, can provide insightful results.

As we mentioned before, our MongoDB was set on a cloud server. However, we didn’t build any data protection method on it, this means any one who knows the IP address can access the DB. Consequently data was stolen twice during the project, which seriously affected our overall development schedule for data collection.

In the future, we would like to apply a Role-based Access Control [9] to our MongoDB system. The main idea of this method is assigning the users one or more roles that determine the user’s access to database resources and operations. Outside of role assignments, the user has no access to the system.

C. Further considerations

Now it is possible to find user's topic distribution, people might be interested whether an particular user always send out tweets about the same subject or their topic changes in time. Besides, maybe there exists a recurring pattern in the variating topic choice. As an easy example, a Twitter maniac might write tweets about barbecues, cocktails and spending days at the beach during summer whereas he might send out thoughts about Christmas, hot chocolate and warm socks during winter months. Such a pattern could occur in terms of days, months, seasons or even years. In other words it's totally worth examining if there is enough accessible data.

On the other hand it might be fascinating to investigate how user's opinions about topics change in time. For instance, people may either have different views of a president's capabilities before and after his term of office or a rather similar one. In this part, the possibilities arise to combine the Topic modelling with the sentiment analysis. However, it will be quite challenging to make both complementary in terms of effectiveness and significance. A possible way setting up this analysis is to calculate the average polarity of the tweets given fixed time slots and compare them to each other. An even bigger challenge would be trying to find anomalies in user's tweet history. A useful method worth examining is the model proposed by Guoxi Zhang, Tomoharu Iwata and Hisashi Kashima [18] that tries to analyze inconsistency in different multi-view text data from different sources.

Finally, in order to increase the accuracy and correctness of the Topic modelling, the choice can be made to feed the model with some background knowledge about the users or topics. The background knowledge of users can be found by using their own given information and possible biography on their profile. To get more knowledge about the topics, news feeds and articles can be tracked and used.

XIV. Conclusion

The empirical results suggest that models implemented in this project perform better, as the dataset grows. Furthermore, LDA model is able to capture a social network phenomenon, regarding a movie release. This shows the power of the data generated by the social network activities. Gathered insights allowed for decision making, for example, when to scale a web application based on peak activity hours.

References

[1] Twitter Usage Statistics Internet Live State 2016

[2] Dave Chaffey. 2018. Smart Insights. [ONLINE] Available at: https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/. [Accessed 4 May 2018].

[3] Alex Hern. 2018. The Guardian. [ONLINE] Available at: https://www.theguardian.com/world/2018/jan/28/fitness-tracking-app-gives-away-location-of-secret-us-army-bases. [Accessed 20 April 2018].

[4] Patrick Greenfield. 2018. The Guardian. [ONLINE] Available at: https://www.theguardian.com/news/2018/mar/26/the-cambridge-analytica-files-the-story-so-far. [Accessed 20 April 2018].

[5] Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[6] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

[7] Godin, F., Vandersmissen, B., De Neve, W. and Van de Walle, R., 2015. Multimedia Lab @ ACL WNUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed Word Representations. In Proceedings of the Workshop on Noisy User-generated Text (pp. 146-153).

[8] https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json

[9] Role-Based Access Control, mongoDB document, https://docs.mongodb.com/manual/core/authorization/

[10] twitter help center, how to use hashtag, https://help.twitter.com/en/using-twitter/how-to-use-hashtags

[11] J. Yang, J. Leskovec. Patterns of Temporal Variation in Online Media. In Proc. Forth ACM International Conference on Web Search and Data Mining (WSDM '11), 2011.

[12] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. The Journal of Machine Learning Research, 3:9931022, 2003

[13] X. Zhao and J. Jiang, An empirical comparison of topics in Twitter and traditional media, Technical Paper Series, School of Information Systems, Singapore Management University, Jan. 2011.

[14] J. Lertsiwaporn and T. Senivongse.Time-Based Visualization Tool for Topic Modeling and Sentiment Analysis of Twitter Messages. 2017

[15] Stevens, Keith, et al. "Exploring topic coherence over many models and many topics." Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Association for Computational Linguistics, 2012.

[16] Dheeraj Rajagopal, Daniel Olsher, Erik Cambria, and Kenneth Kwok. Commonsense-Based Topic Modeling. ACM 978-1-4503-2332-1/13/08, August 2013.

[17] Card, D., Tan, C. and Smith, N.A., 2017. A Neural Framework for Generalized Topic Models. arXiv preprint arXiv:1705.09296.

[18] Zhang, G., Iwata, T. and Kashima, H., 2017, September. Robust Multi-view Topic Modeling by Incorporating Detecting Anomalies. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 238-250). Springer, Cham

[19] Developer.twitter.com. (2018). Rate Limiting. [online] Available at: https://developer.twitter.com/en/docs/basics/rate-limiting [Accessed 7 May 2018].

[20] Benford, F. (1938), The Law of Anomalous Numbers, Proc. Amer. Philosophical Soc. 78, 551572

[21] Durtschi C, Hillison W, Pacini C. The effective use of Benfords law to assist in detecting fraud in accounting data[J]. Journal of forensic accounting, 2004, 5(1): 17-34.

[22] Hill T P. A statistical derivation of the significant-digit law[J]. Statistical science, 1995: 354-363.

[23] Golbeck J. Benfords law applies to online social networks[J]. PloS one, 2015, 10(8): e0135169.

[24] IMDb. (2018). Avengers: Infinity War (2018). [online] Available at: https://www.imdb.com/title/tt4154756/releaseinfo [Accessed 7 May 2018].

[25] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[26] gensim, gensim Topic Modelling for Humans, Available: https://radimrehurek.com/gensim/. [Accessed: May. 4, 2018].