Action rules for sentiment analysis using Twitter

Angelina A. Tzacheva*,
Jaishree Ranganathan and
Arunkumar Bagavathi

Department of Computer Science,
University of North Carolina at Charlotte,
Charlotte, NC, 28223, USA
Email: aatzache@uncc.edu
Email: jrangan1@uncc.edu
Email: abagavat@uncc.edu
*Corresponding author

Abstract: Actionable patterns are interesting and usable knowledge mined from large datasets. Action rules are rules that describe the possible transition of objects from one state to another with respect to a decision attribute. In this work, we extract actionable recommendations in the form of action rules, that can be applied to social networking data in a scalable manner to achieve the desired user goals. We propose extraction of actionable patterns based on sentiment analysis of social network data. In sentiment analysis there are two approaches to determine the polarity: Corpus-based, and Lexicon-based. We use corpus-based sentiment analysis approach, where sentiment values are generated based on sentence structure rather than words. Results show actionable patterns in tweet data and provide suggestions on how to change sentiment of the tweet to a more positive one. The experiment is performed with Twitter data in a distributed environment using Hadoop MapReduce for scalability with large data.

Keywords: sentiment analysis; natural language processing; action rules; MapReduce.

Reference to this paper should be made as follows: Tzacheva, A.A., Ranganathan, J. and Bagavathi, A. (2020) 'Action rules for sentiment analysis using Twitter', *Int. J. Social Network Mining*, Vol. 3, No. 1, pp.35–51.

Biographical notes: Angelina A. Tzacheva is currently teaching as an Associate Professor at the University of North Carolina at Charlotte, USA. Her research interests include data mining, knowledge discovery in databases, distributed knowledge systems, medical imaging, multimedia databases, and bio-informatics.

Jaishree Ranganathan is a PhD student at the University of North Carolina at Charlotte, USA. Her research interests include data mining, text mining, and knowledge discovery in databases, natural language processing, machine learning, and social media mining.

Arunkumar Bagavathi is a PhD student at the University of North Carolina at Charlotte, USA. His research interests include data mining, knowledge discovery in databases, machine learning, and network analysis.

1 Introduction

Social interaction websites like Facebook, Flickr, and Twitter have added a new dimension to the social life of internet-aware people. This trend provides a huge amount of raw data that can be processed to generate structured and useful information. Data mining promises to discover valid and potentially useful patterns in data. Often, discovered patterns are not useful to the user. 'actionability' addresses this problem in that a pattern is deemed actionable if the user can act upon it favourably. Actionable patterns in most cases can be created through rule reduction, model refinement, or parameter tuning by optimising generic patterns (Wang et al., 2006). Actionable patterns are revised optimal versions of generic patterns that capture deeper characteristics and understanding of the business and are also called in-depth or optimised patterns. Action rules are specific patterns extracted from large datasets. To generate action rules, the attributes in the dataset are split into two groups called – flexible attributes and stable attributes. Flexible attributes are those for which the state can change, and the stable attributes are those for which the state is always fixed. An association action rule is a rule extracted from an information system that describes a cascading effect of changes of attribute values listed in the left-hand side of a rule (Ras et al., 2008) on changes of attribute values listed in its right-hand side. Generating action rules based on a classification rules are expensive. This paper makes use of the ARAS algorithm proposed by Ras et al. (2007). ARAS action rules discovery based on grabbing strategy which uses LERS -–combines each action rule generated from single classification rule with the remaining stable attributes to offer more action rules. This work discovers more action rules as compared to the previous algorithms. The use of LERS in the pre-processing module for defining classification rules serves to decrease the complexity of ARAS algorithm. Sentiment analysis is the process of identifying the polarity, opinion or emotion expressed by human. In this work we use the Stanford core NLP suite (Manning et al., 2014) for extracting sentiment from Twitter data.

2 Related work

Many algorithms are available for mining social media data. These data mining algorithms can be divided into several categories. The following are two well established categories: supervised learning and unsupervised learning.

In supervised learning algorithms, the class attributes for datasets are known before running the algorithm. This supervised learning is further classified as classification and regression. When the class attribute is discrete it is classification. Decision tree learning, naive Bayes classifier, and k-nearest neighbour classifier are classification methods. When the class attribute is continuous it is regression. Linear regression and logistic regression are regression methods.

In unsupervised learning, the dataset has no class attribute and the task is to find similar instances and significant patterns in dataset. For example, unsupervised learning can be used to identify events on Twitter data, because the frequency of tweeting is different for various events. This method allows tweets to be grouped based on the times at which they appear and therefore can identify the tweets corresponding to real-world events. This section describes various related research works in this area.

Authors Mishra and Nandi (2015) used Enron e-mail communication dataset to predict the likelihood of future connection between nodes (people) communicating through e-mail where there is no connection between them in present state of the network. This proposal was a Hybrid approach for link prediction which is based on principle of feature ensemble to generate predictive model. Jaccard's coefficient and preferential attachment (JCPA) model, Jaccard's Coefficient, and Adamic/Adar (JCAA) model, preferential attachment, and Adamic/Adar (PAAA) model are generated with the context of ascertaining more accuracy in the performance of link prediction classifier. naive Bayes classifier is the base classifier. The results of this experiment show the proposed hybrid computation method achieves better results with link prediction. The experiment results show that 17,040 pairs of nodes have link labels as connected and label of 170,839 pairs of nodes is unknown which required to be classified. The ratio of disconnected and connected links is 0.099:1. Results are evaluated with confusion matrix and ROC curve.

Authors Hafez et al. (2015) proposed community detection in social network using logic-based programming based on four real social networks – Zachary's karate club network, the bottlenose dolphin network, American college football network and synthetic network – benchmark proposed by Girvan and Newman based on planted partition model. They used deterministic annealing expectation maximisation algorithm in the learning process which yielded promising results when applied to the community detection problem. The model works well with directed and undirected networks and weighted and un-weighted networks. Community in a network is a group of nodes having a high density of edges among the nodes and a low density of edges between different groups. Community detection involves identifying the number of 'k' communities or groups in a network and assigning community membership for each node. The authors Hafez et al. (2015)use PRISM which is a high-level language for probabilistic modelling. Test results on real social networks and synthetic networks were good compared to the NL-EM method results and compared to the original community structure of the networks.

Authors Basuchowdhuri et al. proposed a unified scheme for finding disjoint and overlapping communities in social networks using strength of ties in 2015. Authors proposed a metric that measures strength of a link (SOL) in the network to calculate the degree to which it is part of the community. SOL is formulated based on the number of connections between two adjacent communities and their respective average clustering coefficients to formulate the strength of the links. There are two parts in SOL formulation as follows: part 1 is a calculation of number of connections between the neighbours of the nodes of the link excluding the link. Part 2 is a calculation of average clustering coefficient value of the neighbours of the nodes of the link. Results of this paper show that goodness of a community is defined by its presence of cliques. Structured query language (SQL) based methods work well for graphs with low average shortest path length and high clustering coefficient.

Authors Conan-Guez and Nguyen (2015) proposed an efficient algorithm for hierarchical clustering analysis of large networks called Mod-Mullner. This is a cumulative algorithm for hierarchical clustering analysis. It uses greedy optimisation of modularity, a widely-used measure for network partitioning. Definition of a network community based on the modularity measure 'Q' proposed by Newman (2004) is used to quantify the quality of a given partition 'P' of a network into separate communities. This

work analysed both simulated and real social networks. Mod-Mullner method achieved 1.7 accelerations faster on an average on simulated and real-world networks compared to our fast implementation.

The following studies performed experiments to analyse Twitter data.

Authors Chellal et al. (2016) proposed multi-criterion real time tweet summarisation based upon adaptive method. This method provides new relevant and non-redundant information about an event as soon as it occurs. The tweets selection is based on the following three criterions: informativeness, novelty and relevance with regards of the user interest which are combined as conjunctive condition. Experiments were carried out on TREC MB RTF-2015 (Lin et al., 2015) dataset.

Authors Xu et al. (2016) proposed methods to infer a user's expertise based on their posts on the popular micro-blogging site Twitter. They proposed a sentiment-weighted and topic relation-regularised learning model. Sentiment intensity of a tweet is used to evaluate user's expertise and the relatedness between user's expertises is exploited to model inference problem. The following four common metrics were used for evaluation: accuracy, precision, recall and F1-score.

Authors Bravo-Marquez et al. (2016) proposed a simple model for transferring sentiment labels from words to tweets and vice versa by representing both tweets and words using feature vectors residing in the same feature space. Two transfer learning problems are used to evaluate the approach:

1 training a tweet-level polarity classifier from a polarity lexicon

2 inducing a polarity lexicon from a collection of polarity-annotated tweets.

Tweet centroid model developed in this paper outperformed the classification performance of the popular emoticon-based method for data labelling and better results than a classifier trained from tweets labelled based on the polarity of their words.

Authors Kamkarhaghighi et al. (2016) discovered credible Twitter users in stock market domain. This work suggested a correlation between each user's credibility and the extracted features from each follower network: number of followers; number of stock market-related followers, extracted by a cashtag-based approach; ratio of stock market-related followers to the total number of followers; and the number of seed user tweets.

Authors Bartoli et al. (2016) proposed a language and an inference engine for tweet filtering. This method is an evolutionary approach driven by a multi-objective optimisation scheme to the problem of filtering of Twitter posts. Dataset of Twitter posts authored by 11,254 different Twitter users were assembled by means of an automatic procedure exploiting the Twitter APIs. Egocentric text classifier was used to associate each post with its topics. By Egocentric we mean that class is based on how far the ego is from the ego of that category. Lucia and Ferrari (2014) proposed unsupervised knowledge-based classifier for short text messages where an ego-network represents each category. Classification is based on how far the words are from the ego of that category. Ego-network is built by exploiting YAGO and WordNet. YAGO is a large ontology with high coverage and precision. All objects are represented as entities. WordNet is a lexical database of the English language where words are linked via semantic relationships.

Authors Al-Ayyoub et al. (2015) proposed a lexicon-based sentiment analysis of Arabic tweets. This method is based on sentiment analysis and opinion mining of social network data Twitter feedbacks and comments. Unsupervised approach of sentiment

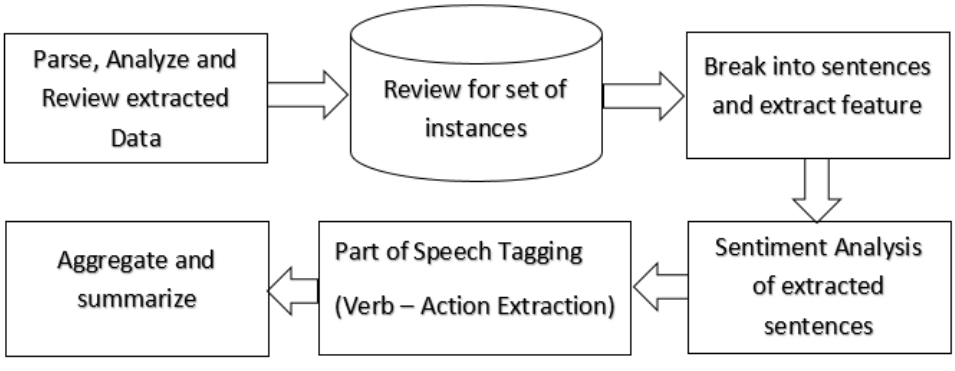
analysis was applied which built a sentiment lexicon sentiment analysis (SA) tool. The sentiment lexicon contains sentiment value for each component of a sentence such as parts of speech (POS), negations, modifiers, clauses, context etc. These values are combined to obtain the sentiment value of entire sentence. This sentiment lexicon was built with about 120,000 Arabic terms and a SA tool based on predicate calculus.

In this work, we propose a new method of discovering actionable patterns through action rules for the Twitter data with sentiment analysis (Manning et al., 2014) results. We are not aware of any other research that analyse tweets using action rules.

3 Methodology

The focus of this work is to mine actionable patterns via action rules from Twitter data and provide suggestions on how users can be more positive. The proposed method shown in Figure 1 consists of the following components: data collection, pre-processing, sentiment analysis, classification, action rule generation, and summarisation.

Figure 1 Actionable pattern mining system for twitter sentiment analysis



3.1 Data collection

We collected data using Twitter standard search API. The following attributes are extracted as part of the data collection process: RetweetCount, IsFavorited, UserID, UserFriendsCount, UserFavoritesCount, UserFollowersCount, UserLanguage and TweetText. We collected close to 28,000 instances for the experiments purpose. Sample data shown in Table 3.

3.2 Pre-processing

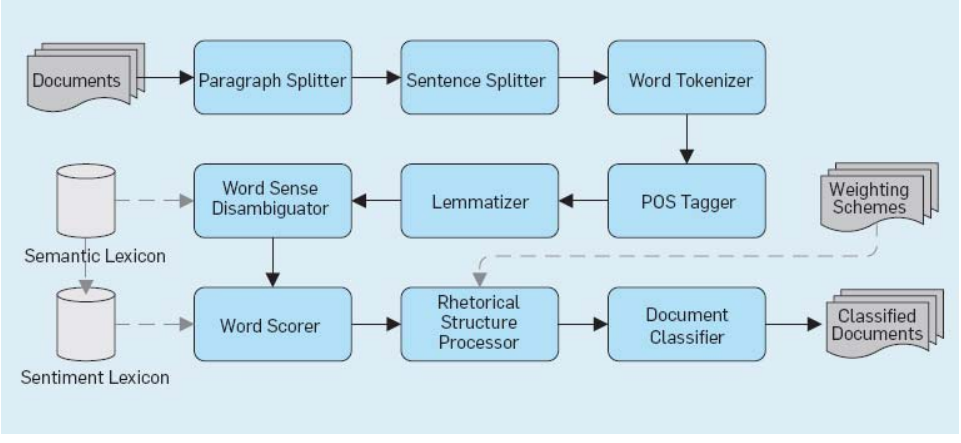
Pre-processing is discretisation on the attributes UserFriendsCount, UserFavoritesCount, and UserFollowers Count by placing them in intervals. Intervals are selected based on the range of friends, favourites, and followers count in the actual data, for instance if the user's friends count is within 100 then we discretise them as 0–100. In this step, we cleaned the data from missing values, applied feature selection and removed unnecessary values. In pre-processing step, we aim to represent the data in a form that can be analysed

efficiently and to improve their quality by reducing the amount of trivial noise. We kept the following attributes: RetweetCount, IsFavorited, UserID, UserFriendsCount, UserFavoritesCount, UserFollowersCount, UserLanguage, and TweetText.

3.3 Sentiment analysis

We performed Sentiment Analysis on data and augmented the data with class attribute which is 'Sentiment' that has the following values: positive, negative, neutral, very positive, and very negative. In addition, we extracted Parts of Speech and augmented the data with attribute verb. Example for Parts of Speech is shown in Figure 3. This information is required in actionable pattern mining because verbs suggest actionable knowledge.

Figure 2 Sentiment analysis (see online version for colours)



Stanford core NLP (Manning et al., 2014) was used for sentiment analysis. The general process for sentiment analysis is shown in Figure 2. This NLP suite provides a set of natural language analysis tools. The basic distribution provides model files for the analysis of well-edited English, but the engine is compatible with models for other languages. Stanford core NLP is written in Java (Manning et al., 2014). This NLP suite provides various annotators which can work with any character encoding, making use of Java's Unicode support, but system defaults to UTF-8 encoding. Out of these annotators we are using tokeniser, part of speech, and sentiment analysis in our work.

Figure 3 Part-of-speech tagger – verbs

Input: Grab your referral link today!

POS Tagger: Grab/VB your/PRP$ Referral/NN Link/NN today/NN !/.

3.4 Classification

We use learning from examples using rough sets (LERS) (Grzymala-Busse et al., 2013) algorithm to extract classification rules from Twitter data. Each tweet was classified as

positive, negative, neutral, very positive, very negative. LERS (Grzymala-Busse et al., 2013) is used to extract classification rules from the information system. Our implementation follows distributed strategy of generating classification rules using LERS system shown in Figure 4. Using the information system S from Table 1, LERS strategy generates certain and possible rules describing decision attribute D in terms of attributes A, B, and C. LERS can be used as a data strategy to generate classification rules. LERS produces a set of certain and possible rules (Grzymala-Busse et al., 2013). We consider only marked certain rules to construct action rules. Since LERS follows bottom-up strategy, it constructs rules with a conditional part of length x, then it continues to construct rules with a conditional part of length $x + 1$ during the following iterations.

Figure 4 LERS algorithm

Algorithm 1:

LERS (*attributesSupport*, *decisionSupport*)

(where *attributesSupport* and *decisionSupport* are maps with distinct attribute values as keys and their corresponding value is the objects in the information system supporting them)

fixedSupport ← *attributesSupport*
while *attributesSupport* is not empty **do**
  for each key, value pair in the *attributeSupport* **do**
    if value is a subset of one of the values of *decisionSupport* **then**
      Add key and *decisionValue* to *certainRules*(where *certainRules* is a map with attribute value as a 'key' and decision attribute value as a 'value')
    else
      Add key and value to *possibleRules*
      (where *possibleRules* is a map with attribute value as a 'key' and decision attribute value as a 'value')
    end
    delete key from the *attributesSupport*
  end
  for each key_1, $value_1$ pair in the *possibleRules* **do**
    for each key_2, $value_2$ pair in the *fixedSupport* **do**
      if key_1 contains key_2 then Continue
      else
        $key_3 \leftarrow (key_2, key_1)$
        $value_3 \leftarrow$ Set of objects in information system supporting key3
        Add key_3 and $value_3$ to *attributeSupport*
      end
    end
  end
end

Table 1 Example information system

X	A	B	C	D
x_1	a_1	b_1	c_1	d_1
x_2	a_3	b_1	c_1	d_1
x_3	a_2	b_2	c_1	d_2
x_4	a_2	b_2	c_2	d_2
x_5	a_2	b_1	c_1	d_1
x_6	a_2	b_2	c_1	d_2
x_7	a_2	b_1	c_2	d_2
x_8	a_1	b_2	c_2	d_1

For the information system given in Table 1, consider the following as decision support:

$$(d_1) - * = \{x_1, x_2, x_5, x_8\}; (d_2) * = \{x_3, x_4, x_6, x_7\}$$

LERS module given in Figure 4 for the given information system S, extracts certain and possible rules which are given in Table 2.

Table 2 LERS example for information system S

Iteration	*Attribute value support*	*Certain rules*	*Possible rules*
1	$(a_1) * = \{x_1, x_8\}$ – marked	$a_1 \rightarrow d_1$	$a_2 \rightarrow d_1$
	$(a_2) * = \{x_3, x_4, x_5, x_6, x_7\}$	$a_3 \rightarrow d_1$	$a_2 \rightarrow d_2$
	$(a_3) * = \{x_2\}$ – marked		$b_1 \rightarrow d_1$
	$(b_1) * = \{x_1, x_2, x_5, x_7\}$		$b_1 \rightarrow d_2$
	$(b_2) * = \{x_3, x_4, x_6, x_8\}$		$b_2 \rightarrow d_1$
	$(c_1) * = \{x_1, x_2, x_3, x_5, x_6\}$		$b_2 \rightarrow d_2$
	$(c_2) * = \{x_4, x_7, x_8\}$		$c_1 \rightarrow d_1$
			$c_1 \rightarrow d_2$
			$c_2 \rightarrow d_1$
			$c_2 \rightarrow d_2$
2	$(a_2, b_1) * = \{x_5, x_7\}$	$a_2 \wedge b_2 \rightarrow d_2$	$a_2 \wedge b_1 \rightarrow d_1$
	$(a_2, b_2) * = \{x_3, x_4, x_6\}$ – marked	$a_2 \wedge c_2 \rightarrow d_2$	$a_2 \wedge b_1 \rightarrow d_2$
	$(a_2, c_1) * = \{x_3, x_5, x_6\}$	$b_1 \wedge c_1 \rightarrow d_1$	$a_2 \wedge c_1 \rightarrow d_1$
	$(a_2, c_2) * = \{x_4, x_7\}$ – marked	$b_1 \wedge c_2 \rightarrow d_2$	$a_2 \wedge c_1 \rightarrow d_2$
	$(b_1, c_1) * = \{x_1, x_2, x_5\}$ – marked	$b_1 \wedge c_2 \rightarrow d_2$	$b_2 \wedge c_2 \rightarrow d_1$
	$(b_1, c_2) * = \{x_7\}$ – marked		$b_2 \wedge c_2 \rightarrow d_2$
	$(b_2, c_1) * = \{x_3, x_6\}$ – marked		
	$(b_2, c_2) * = \{x_4, x_8\}$		
3	$(a_2, b_1, c_1) * = \{x_5\}$ – marked	$a_2 \wedge b_1 \wedge c_1 \rightarrow d_1$	

3.5 Actionable pattern mining – action rules

ARAS is action rules discovery based on agglomerative strategy, which uses LERS proposed by Ras et al. (2007) as an alternative to system DEAR (Ras and Tsay, 2003) which extracts action rules from a pair of classification rules. The foremost advantage of using ARAS is that it uses single classification rule to provoke action rules. ARAS uses an algorithm like LERS to extract action rules, without the need of verifying the validity of the certain relations. The algorithm checks if these relations are marked previously by LERS.

Figure 5 AR (action rules) algorithm in distributed environment using MapReduce

Algorithm 2:

AR (*certainRules*, *decisionFrom*, *decisionTo*)

(where *certainRules* is provided by the LERS)
for each *key*, *value* pair in the *certainRules* **do**
if $value_1$ **equals** *decisionTo* **then**
actions ← empty list
for each attribute value *a* in *key* **do**
A←attributeName(*a*)
actions. Add ("($A, \rightarrow a$)")
end

ARAS presumes that system LERS construct classification rules describing target decision value. Figure 4 and Figure 5 together gives the algorithm of ARAS. Algorithm AR takes each candidate classification rule and form an action rule schema which in turn is given to the algorithm ARAS to build a cluster of action rules around each schema. For the classification rules in Table 2, algorithm AR generates following set of action rule schema:

$$AR_{s1}(d_1 \rightarrow d_2) = (A, \rightarrow a_2) \wedge (B, \rightarrow b_2) \rightarrow (D, d_1 \rightarrow d_2)$$

$$AR_{s2}(d_1 \rightarrow d_2) = (A, \rightarrow a_2) \wedge (C, \rightarrow c_2) \rightarrow (D, d_1 \rightarrow d_2)$$

$$AR_{s3}(d_1 \rightarrow d_2) = (B, \rightarrow b_2) \wedge (C, c_2) \rightarrow (D, d_1 \rightarrow d_2)$$

$$AR_{s4}(d_1 \rightarrow d_2) = (B, \rightarrow b_2) \wedge (C, \rightarrow c_1) \rightarrow (D, d_1 \rightarrow d_2)$$

Algorithm *ARAS* takes each action rule schema and using their flexible and stable attributes generates following action rules which imply $d_1 \rightarrow d_2$. For the action rule schema AR_{s1}, the algorithm *ARAS* finds all missing flexible attributes A_M: $\{a_1, a_3, b_1\}$. Each missing flexible attribute is filled into appropriate action terms. In ARAS, the maximum number of action rules generated = A_M. For AR_{s1}, ARAS produces following action rules:

$$AR_1(d_1 \to d_2) = (A, a_1 \to a_2) \wedge (B, \to b_2) \rightarrow (D, d_1 \to d_2)$$

$$AR_2(d_1 \to d_2) = (A, a_3 \to a_2) \wedge (B, \to b_2) \rightarrow (D, d_1 \to d_2)$$

Figure 6 ARAS in a distributed environment using MapReduce

Algorithm 3:
ARAS (*actions*, *decisionFrom*, *decisionTo*)

(where 'actions' is a list of actions from Algorithm AR)
stableValues ← list of stable attribute values in *actions*
actionsSupport ←set of objects in the information system supporting *stableValues* ∩ *decisionFrom*
missingValues ←set of missing flexible attribute values of the flexible attributes in actions
for each *value* in *missingValues* **do**
  newValues ← combine value with *stableValues*
  newSupport ← set of objects in the information system supporting *newValues* in actions
  if *newSupport* ⊆ *actionsSupport* **then**
    Add *value* to *actions*
    Output *actions* as Action Rule
  end
end

Let an action rule R takes a form of:

$$(Y_1 \to Y_2) \rightarrow (Z_1 \to Z_2)$$

where,

Y is the condition part of R

Z is the decision part of R

Y_1 is a set of all left side of the all condition action terms

Y_2 is a set of all right side of the all condition action terms

Z_1 is the decision attribute value on left side

Z_2 is the decision attribute value on right side

In Ras et al (2007), the support and confidence of an action rule R is given as

$$\text{Support(R)}=\min\left\{\text{card}(Y_1 \cap Z_1), \text{card}(Y_2 \cap Z_2)\right\}$$

$$\text{Confidence(R)} = \left[\text{card}(Y_1 \cap Z_1)/\text{card}(Y_1)\right] * \left[\text{card}(Y_2 \cap Z_2)/\text{card}(Y_2)\right]$$

In this paper, we use the following support and confidence formula given by Tzacheva et.al. (2016b) to reduce the complexity.

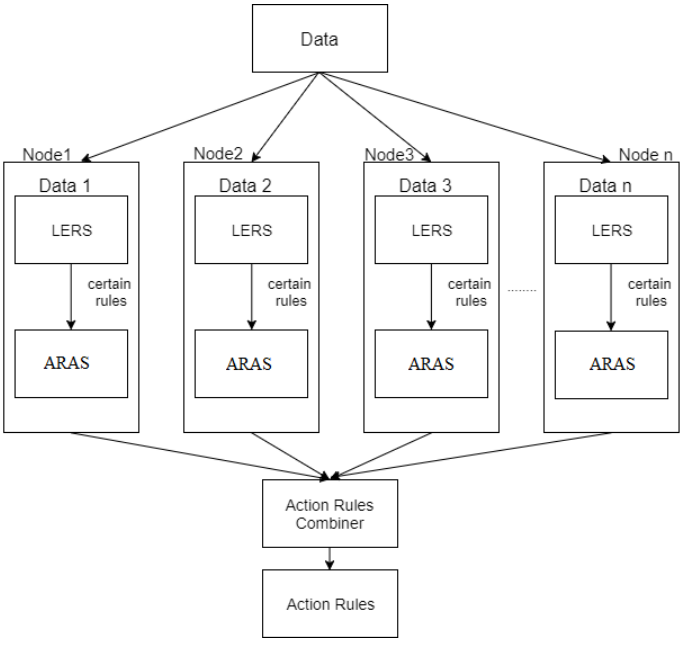
$$\text{Support(R)} = \text{card}(Y_2 \cap Z_2)$$

$$\text{Confidence(R)} = \left[\text{card}(Y_2 \cap Z_2)/\text{card}(Y_2)\right]$$

3.6 Distributed actionable pattern mining – random-forest Hadoop

We use MR – random-forest algorithm for distributed action rules discovery by authors Tzacheva et al. (2016a), using Apache Hadoop framework and Google MapReduce (Dean and Ghemawat,2008). MR-Random-Forest algorithm is shown in Figure 6. We take as an input a set of files: the data, the attribute names, user specified parameters such as: minimum support, and confidence thresholds, stable attribute names, flexible attribute names, decision attribute choice, decision attribute value to change from, and decision attribute value to change to, which is the desired value of decision attribute (desired object state). We import these input files into the Hadoop distributed file system (HDFS). Action rules are built using Apache framework and results are evaluated using Hadoop MapReduce system. The data is spread across a distributed environment to get more optimal action rules and upgraded ARAS algorithm to get more specific action rules. Figure 7 shows the overview of the algorithm used for generating action rules.

Figure 7 MR – random forest algorithm for distributed action rules discovery



4 Experiment and results

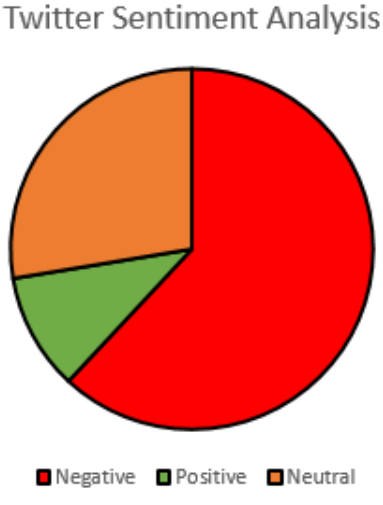
Our research is focussed on recommending methods to improve emotions from negative to positive and neutral to positive and increasing friends count of a user. For this experiment, we used live tweets extracted using Twitter Search API on the latest tweets. The Twitter Search API searches against a sampling of recent tweets published in the past seven days. Data Collection includes user-generated updates collected directly from social media API as they allow subscription to a continuous live stream of data. Our data contains the following attributes: RetweetCount, IsFavorited, UserID, UserFriendsCount,

UserFavoritesCount, UserFollowersCount, TweetText, UserLanguage, TweetSentiment, and TweetVerb. We collected about 28,000 instances. Table 3 gives the description about the dataset. The Hadoop research cluster at University of North Carolina Charlotte was used to perform the experiments. This cluster has six nodes connected via ten gigabits per second ethernet network.

Table 3 Sample data with sentiment analysis results

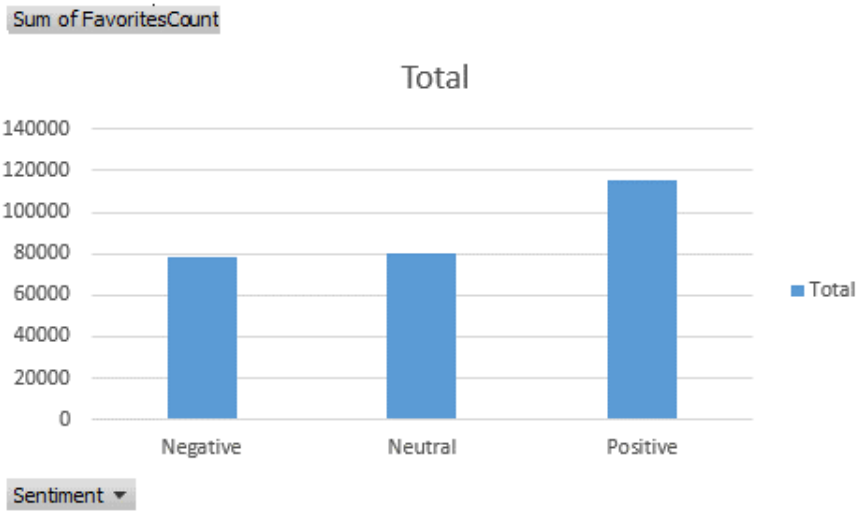
ReTweet	*IsFavorited*	*Friends*	*Followers*	*Language*
0	False	247	30,795	en
0	False	42	527	13
	Text		*Sentiment*	*Verb*
0	RT @ThadSuggs: PLS HELP.SAVEA GRADE.		Negative	Help
0	RT @CW_TheFlash: Thanks for watching! New episodes return January 24.		Positive	watching

Figure 8 Tweets sentiment analysis (see online version for colours)



We used action rules to change the emotion from negative to positive and neutral to positive, also to change from lower number of friends count to higher number of friends. Our data contains the following attributes: RetweetCount, IsFavorited, UserID, UserFriendsCount, UserFavoritesCount, UserFollowersCount, TweetText, UserLanguage, TweetSentiment, and TweetVerb. In Figure 8 we can see that most of the comments are negative. We focus on the issue of improving the comments emotion from negative to positive and neutral to positive by providing actionable patterns to improve the emotions. Also, we suggest actionable patterns to improve friends count. In our data, we noticed that positive comments have more favourites compared to others as shown in Figure 9.

Figure 9 Favourites count for various sentiments (see online version for colours)



The methods suggested in previous section LERS, ARAS for classification and action rules mining were implemented in Java and tested with the above data set.

Let us consider AR1 from Table 5. If user's favourites count increases from 0–100 to 1,001–5,000 and user's followers count increases from 101–200 to 701–800 and user language is English, then tweet sentiment could be changed from neutral to positive. This rule is generated with a confidence of 100% and support 2 for the Twitter data with following attributes: RetweetCount, IsFavorited, UserID, UserFriendsCount, UserFavoritesCount, UserFollowersCount, UserLanguage and TweetSentiment. In future if more attributes relevant to the context of text like frequency of part-of-speech including adjectives is added then we anticipate that the action rules generated by our system would be more intuitive.

Table 4 Single node and Hadoop cluster time comparison

Experiment	*Time taken single node*	*Time taken Hadoop*
Experiment 1	432 seconds	258 seconds
Experiment 2	270 seconds	180 seconds
Experiment 3	273 seconds	192 seconds

Considering the recent growth of the amount of data collected nowadays, we use distributed implementation of the proposed method LERS and ARAS using Hadoop map reduce framework by Tzacheva et al. (2016a). We show that computation is much faster in the distributed framework than on single computer. The experiment results are shown in Table 4. We can scale to large social media data sizes. It is considered that the workload can be spread across two nodes and can increase the number of mappers and scale to very large size data and handle it appropriately.

Table 5 Example action rules generated

Action rule no.	*Action rule*
AR1	(UserFavoritesCount, 0–100 → 1,001–5,000) ^ (UserFollowersCount, 101–200 → 701–800) ^ (UserLanguage = en-gb) ➔ (TweetSentiment, Neutral → Positive) [Support: 2, Old Confidence: 66%, New Confidence: 100%]
AR2	(UserFavoritesCount, 101–200 → 701–800) ^ (UserFollowersCount, 0–100 → 301–400) ^ (UserFriendsCount, 101-200 → 201–300) ^ (UserLanguage = en) ➔ (TweetSentiment, Neutral → Positive) [Support: 2, Old Confidence: 80%, New Confidence: 100%]
AR3	(UserFavoritesCount, 201–300 → 601–700) ^ (UserFollowersCount, 5,001–10,000 → 301–400) ^ (UserFriendsCount, 501–600 → 701–800) ➔ (TweetSentiment, Neutral → Positive) [Support: 2, Old Confidence: 100%, New Confidence: 100%]

Experiment 1: UserFriendsCount, following attributes were used to generate action rules: decision attribute – UserFriendsCount, stable attribute – UserLanguage, support – 2, confidence – 60%. Sample action rules generated for this experiment are given in Table 6.

Table 6 Example action rules – experiment 1: change from class UserFriendsCount: low to high number of friends

Single node action rules	*Hadoop action rules*
(TweetSentiment, negative → positive) ^ (UserFavoritesCount, 0–100 → 10,001–15,000) ^ (UserFollowersCount, 0–100 → 1,001–5,000) ^ (UserLanguage = pt) ➔ (UserFriendsCount, 0–100 → 1,001–5,000) [support: 2, old confidence: 67%, new confidence: 100%]	(TweetSentiment, negative → neutral) ^ (UserFavoritesCount, 0–100 → 5,001–10,000) ^ (UserLanguage = it) ➔ (UserFriendsCount, 0–100 → 1,001–5,000) [support: 2, old confidence: 60%, new confidence: 100%]
(TweetSentiment, Neutral → Positive) ^ (UserFavoritesCount, 0–100 → 10,001–15,000) ^ (UserFollowersCount, 0–100 → 1,001–5,000) ^ (UserLanguage = pt) ➔ (UserFriendsCount, 0–100 → 1,001–5,000) [support: 2, old confidence: 76%, new confidence: 100%]	(TweetSentiment, neutral → positive) ^ (UserFavoritesCount, 601–700 → 1,001–5,000) ^ (UserLanguage = de) ➔ (UserFriendsCount, 0–100 → 1,001–5,000) [support: 2, old confidence: 100%, new confidence: 100%]
(TweetSentiment, negative → neutral) ^ (UserFavoritesCount, 20,001–25,000 → 601–700) ^ (UserFollowersCount, 0–100 → 5,001–10,000) ^ (UserLanguage = en) ➔ (UserFriendsCount, 0–100 → 1,001–5,000) [support: 4, old confidence: 96%, new confidence: 100%]	(TweetSentiment, very negative → neutral) ^ (UserFavoritesCount, 0–100 → 601–700) ^ (UserFollowersCount, 0–100 → 5,001–10,000) ^ (UserLanguage = en) ➔ (UserFriendsCount, 0–100 → 1,001–5,000) [support: 2, old confidence: 100%, new confidence: 100%]

Experiment 2: negative to positive, following attributes were used to generate action rules: decision attribute – TweetSentiment, stable attribute – UserLanguage, support – 2, confidence – 60%. Sample action rules generated for this experiment are given in Table 7.

Table 7 Example action rules – experiment 2: change class from TweetSentiment: negative to positive

Single node action rules	*Hadoop action rules*
(UserFavoritesCount, 10,001–15,000 → 601–700) ^ (UserFollowersCount, 101–200 → 301–400) ^ (UserFriendsCount, 301–400 → 701–800) ➔ (TweetSentiment, negative → positive) [support: 2, old confidence: 60%, new confidence: 100%]	(UserFavoritesCount, 0–100 → 601–700) ^ (UserFollowersCount, 5,001–10,000 → 301–400) ^ (UserFriendsCount, 201–300 → 701–800) ➔ (TweetSentiment, negative → positive) [support: 2, old confidence: 100%, new confidence: 100%]
(UserFavoritesCount, 101–200 → 15,001–20,000) ^ (UserFollowersCount, 101–200 → 701–800) ^ (UserLanguage = de) ➔ (TweetSentiment, negative → positive) [support: 2, old confidence: 100%, new confidence: 100%]	(UserFavoritesCount, 0–100 → 601–700) ^ (UserFollowersCount, 30,000-above → 301–400) ^ (UserFriendsCount, 201–300 → 701–800) ➔ (TweetSentiment, negative → positive) [support: 2, old confidence: 100%, new confidence: 100%]
(UserFavoritesCount, 501–600 → 601–700) ^ (UserFollowersCount, 101–200 → 301–400) ^ (UserFriendsCount, 801–900 → 701–800) ➔ (TweetSentiment, negative → positive) [support: 2, old confidence: 100%, new confidence: 100%]	(UserFollowersCount, 5,001–10,000 → 201–300) ^ (UserFriendsCount, 5,001–10,000 → 0–100) ^ (UserLanguage = es) ➔ (TweetSentiment, negative → positive) [support: 3, old confidence: 75%, new confidence: 100%]

Experiment 3: neutral to positive, following attributes were used to generate action rules: decision attribute – TweetSentiment, stable attribute – UserLanguage, support – 2, confidence – 60%. Sample action rules generated for this experiment are given in Table 8.

Table 8 Example action rules – experiment 3: change class from TweetSentiment: neutral to positive

Single node action rules	*Hadoop action rules*
(UserFavoritesCount, 30,000–above → 601–700) ^ (UserFollowersCount, 20,001–25,000 → 301–400) ^ (UserFriendsCount, 101–200 → 701–800) ➔ (TweetSentiment, neutral → positive) [support: 2, old confidence: 66%, new confidence: 100%]	(UserFavoritesCount, 30,000–above → 601–700) ^ (UserFollowersCount, 20,001–25,000 → 301–400) ^ (UserFriendsCount, 101–200 → 701–800) ➔ (TweetSentiment, neutral → positive) [support: 2, old confidence: 66%, new confidence: 100%]
(UserFavoritesCount, 101–200 → 601–700) ^ (UserFollowersCount, 901–1,000 → 301–400) ^ (UserFriendsCount, 201–300 → 701–800) ➔ (TweetSentiment, Neutral → Positive) [Support: 2, Old Confidence: 66%, New Confidence: 100%]	(UserFavoritesCount, 101–200 → 601–700) ^ (UserFollowersCount, 901–1,000 → 301–400) ^ (UserFriendsCount, 201–300 → 701–800) ➔ (TweetSentiment, neutral → positive) [support: 2, old confidence: 100%, new confidence: 100%]
(UserFavoritesCount, 30,000–above → 601–700) ^ (UserFollowersCount, 30,000–Above → 301–400) ^ (UserFriendsCount, 201–300 → 701–800) ➔ (TweetSentiment, neutral → positive) [support: 2, old confidence: 100%, new confidence: 100%]	(UserFavoritesCount, 5,001–10,000 → 701–800) ^ (UserFollowersCount, 1,001–5,000 → 301–400) ^ (UserFriendsCount, 101–200 → 201–300) ^ (UserLanguage = en) ➔ (TweetSentiment, neutral → positive) [support: 2, old confidence: 77%, new confidence: 100%]

5 Conclusions

This work proposed a new approach to analyse sentiment of tweets through mining actionable patterns via action rules. We suggest actions that can be undertaken to reclassify user sentiment from negative to positive and neutral to positive using comments. We also suggest actions of how users can increase their friends count. We provide implementation on both single machine and cloud distributed environment for scalability purpose. We compare the results with single machine implementation and distributed Hadoop MapReduce framework. Our experiments show that the processing of the proposed algorithm runs faster on distributed environment than on single machine. The proposed method can scale to accommodate large social media data size. Future work includes augmenting the data set with more syntactical parts including nouns and adjectives and to build lexicons for specific subjects. For example, financial, medical, and industrial topics.

References

Al-Ayyoub, M., Essa, S.B. and Alsmadi, I. (2015) 'Lexicon-based sentiment analysis of arabic tweets', *International Journal of Social Network Mining*, Vol. 2, No. 2, pp.101–114.

Bartoli, A., Carminati, B., Ferrari, E. and Medvet, E. (2016) 'A language and an inference engine for Twitter filtering rules', in *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'16)*, IEEE 2016, Omaha, USA, October, pp.614–617.

Basuchowdhuri, P., Prabhu, V.L., Roy, M., Majumder, S. and Saha, S.K. (2015) 'Unified scheme for finding disjoint and overlapping communities in social networks using strength of ties', *International Journal of Social Network Mining*, Vol. 2, No. 2, pp.173–202.

Bravo-Marquez, F., Frank, E. and Pfahringer, B. (2016) 'From opinion lexicons to sentiment classification of tweets and vice versa: a transfer learning approach', in *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'16*), IEEE 2016, Omaha, USA, October, pp.145–152.

Chellal, A., Boughanem, M. and Dousset, B. (2016) 'Multi-criterion real time tweet summarization based upon adaptive threshold', in *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'16)*, IEEE 2016, Omaha, USA, October, pp.264–271.

Conan-Guez, B. and Nguyen, M.C. (2015) 'Mod-Mullner: an efficient algorithm for hierarchical community analysis in large networks', *International Journal of Social Network Mining*, Vol. 2, No. 2, pp.133–157

Dean, J. and Ghemawat, S. (2008) 'MapReduce: simplified data processing on large clusters', *Communications of the ACM*, Vol. 51, No. 1, pp.107–113.

Grzymala-Busse, J.W., Marepally, S.R. and Yao, Y. (2013) 'An empirical comparison of rule sets induced by LERS and probabilistic rough classification', in *Rough Sets and Intelligent Systems – Professor Zdzislaw Pawlak in Memoriam*, Springer, Berlin, Heidelberg, pp.61–276.

Hafez, A.I., Al-Shammari, E.T., Hassanien, A.E. and Fahmy, A.A. (2015) 'Community detection in social networks using logic-based probabilistic programming', *International Journal of Social Network Mining*, Vol. 2, No. 2, pp.158–172.

Kamkarhaghighi, M., Chepurna, I., Aghababaei, S. and Makrehchi, M. (2016) 'Discovering credible Twitter users in stock market domain', in *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'16)*, IEEE 2016, Omaha, USA, October, pp.66–72.

Lin, J., Efron, M., Sherman, G., Wang, Y. and Voorhees, E.M. (2015) 'Overview of the TREC 2015 microblog track', in *Proceedings of Text Retrieval Conference, (TREC2015)*, Gaithersburg, USA, 17–20 November.

Lucia, W. and Ferrari, E. (2014) 'Egocentric: ego networks for knowledge-based short text classification', in *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, November, pp.1079–1088.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. and McClosky, D. (2014) 'The Stanford CoreNLP natural language processing toolkit', in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, USA, pp.55–60.

Mishra, S. and Nandi, G.C. (2015) 'A novel hybrid approach for link prediction problem in social network', *International Journal of Social Network Mining*, Vol. 2, No. 2, pp.115–132.

Newman, M.E. (2004) 'Detecting community structure in networks', *The European Physical Journal B (EPJ B)*, Vol. 38, No. 2, pp.321–330.

Ras, Z.W. and Tsay, L.S. (2003) 'Discovering extended action-rules (system DEAR)', in *Proceedings of Intelligent Information Processing and Web Mining*, Springer, Berlin, Heidelberg, pp.293–300.

Ras, Z.W., Dardzinska, A., Tsay, L.S. and Wasyluk, H. (2008) 'Association action rules', in *Proceedings of IEEE International Conference on Data Mining Workshops*, *(ICDMW'08)*, pp.283–290.

Ras, Z.W., Wyrzykowska, E. and Wasyluk, H. (2007) 'ARAS: action rules discovery based on agglomerative strategy', in *Proceedings of 2007 International Workshop on Mining Complex Data*, Springer, Berlin, Heidelberg, September, pp.96–208.

Tzacheva, A.A., Bagavathi, A. and Ganesan, P.D. (2016a) 'MR-random forest algorithm for distributed action rules discovery', *International Journal of Data Mining and Knowledge Management Process (IJDKP)*, Vol. 6, No. 5, pp.15–30.

Tzacheva, A.A., Sankar, C.C., Ramachandran, S. and Shankar, R.A. (2016b) 'Support confidence and utility of action rules triggered by meta-actions', in *IEEE International Conference on Knowledge Engineering and Applications (ICKEA2016)*, September, Singapore, pp.113–120.

Wang, K., Jiang, Y. and Tuzhilin, A. (2006) 'Mining actionable patterns by role models', in *Proceedings of 22nd IEEE International Conference on Data Engineering (ICDE'06)*, Atlanta, USA, April, pp.16–21.

Xu, Y., Zhou, D. and Lawless, S. (2016) 'Inferring your expertise from twitter: integrating sentiment and topic relatedness', In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'16)*, IEEE 2016, Omaha, USA, October, pp.121–128.