CAN TWITTER BE A LEGITIMATE INTELLIGENCE TOOL?

AUTHOR

Harshit Aggarwal is a third year college student at Drexel University. Pursuing a degree in computer engineering, he hopes to graduate with a Bachelor's of Science degree in June 2016. With a penchant for math and science, he has become proficient in several programming languages and platforms including R, C++, MATLAB and MySQL.

Harshit has been a co-op/intern with CSC's ResearchNetwork for six months and has focused on investigating Twitter as a competitive intelligence tool.

You can contact Harshit via LinkedIn at: https://www.linkedin.com/in/HarshitAggarwal1.

INTRODUCTION

Relationships have always been at the heart of marketing. For decades, consumers have trusted individuals-friends, family for reliable information about products and services. In today's digital world, relationships remain a critical factor in discovering and evaluating potential purchases. The key to make sales in the modern world is to make a connection with the buyer and build their trust; this is known as social selling. One way to listen to the potential customers is through the immensely popular microblogging website Twitter.

Twitter is used by most celebrities, politicians, journalists and potential customers and is at the helm of social interaction. The technique of listening in to customers through social media is known as social listening. Social networks are the ultimate sales networking tool because you can meet so many people so quickly. Better use of social network data could provide CSC with opportunities to open sales since social networks are perfect platforms for initiating conversations.

CSC could increase their outreach to a wider range of customers. CSC can connect to people not only to drive sales but also to make connections with them. Making a connection with the customer can also lead to word-of-mouth marketing. A satisfied customer/individual, with whom CSC has established a connection, is more likely to recommend CSC products to their colleagues, family and friends. For example, the Drake hotel in Chicago, replied to a tweet by famous blogger David Armano about the weather and with something so small they began making a personal connection with their customer. They studied his tweets/blogs to get to know him on a personal level and enhanced their engagement with him.

After Armano confirmed a room in the hotel, they replied by tweeting "I'll alert the valet to start practicing their Harley parking skills," showing that they know Armano on a personal level. Following this Armano tweeted "Nice engagement via @DrakeChicago. Twitter account as concierge" which reached out to thousands of Armano's followers (Martin). Corporations such as Microsoft have been actively listening in on their customers on Twitter and been using it to offer promotions and support. This leads to the question about whether Twitter can be used for information besides support and promotions. In particular the ability of social media such as Twitter to provide competitive intelligence.

According to Digimind's Marketing intelligence survey report:

- 69.4% companies monitor LinkedIn
- 62.5% companies monitor Twitter
- 47.2% companies monitor Facebook

- 35.2% companies monitor Google+

Further the author of the survey dicovered that "...there is a tendency for some people to view social media monitoring as something which solely benefits B2C environments but it's fast being adopted by a wide range of businesses and organizations for unearthing valuable intelligence insights (Keefe)." Some companies have been tactically using social media to keep the customers happy and increase their outreach. Organizations such as NASA are using Twitter to reach users beyond their followers. This activity doesn't need to be fancy. Something simple like engaging with a popular account in a playful way can help to increase mindshare. According to the social media manager at NASA: "We try to engage influencers. We tweeted a happy birthday to Star Trek's Leonard Nimoy, engaged with Tom Hanks reliving the making of the film Apollo 13 and were also engaged and got retweeted by Justin Bieber." Simply by engaging with Justin Bieber, NASA reached out to over 61 million followers of his Twitter account.

This shows how companies can use social media for more than just promotions, they can use it to reach new customers, increase awareness of their products or make a meaningful connection.

RELATED WORK

Tools such as Brandwatch and Adobe Social already offer similar services and exploring them further was the next logical step. Brandwatch is a social media monitoring company which uses crawlers to search through social media websites and bring out relevant information. They have the ability to crawl through 70 million websites in 27 different languages. They have a specific algorithm that checks whether the content on the website is real and not navigational text or adverts. An example of one of their cases can be found on their website.

The company was hired by ESPN to capitalize on the excitement and enthusiasm surrounding the 2014 World Cup and drive fans towards their website. ESPN partnered with the Transport for London (TFL) “to display game results on announcement boards at 150 stations across London.” By using Brandwatch they tracked the progress of the campaign which “sparked engagement with the key demographic” and “achieved over 60% positive mentions about the campaign.” After the campaign Charles Boss, the head of marketing for ESPN UK said: “Thanks to this innovative and well-managed campaign, unique visitors to ESPN FC’s website increased by 25% year on year.” (Brandwatch)

Another product is Adode Social, which is a part of their Adobe Marketing Cloud. Adobe Social is a tool for managing social content and social campaigns. It's a comprehensive solution for building stronger connections through data-driven content. It deals with relevant posts, insightful conversations, measurable results, and social activities connected to business. It allows users to “define, assign, and control access so that users always see the right tools and data” as well as “monitor the pulse of social conversations for trends, opportunities, and potential business threats. Gauge the health of brands, track customer sentiment, and measure brand share of voice compared to competitors.” It allows content to be published on multiple social media websites through the application, so content does not have to be recreated for each social media website.

An interesting feature of Adobe Social is its “predictive publishing” functionality. With predictive publishing: “The platform will be able to predict the amount of engagement a

Tweet, Facebook post, or any piece of social content will receive before it’s published. It examines the crafted content and measures it against previous posts and will offer up suggestions on not only how to improve it, but the best times to share it with your community.” Within the platform Adobe Social also allows users to “create and deploy custom engagement experiences like contests, galleries, polls, and coupons for Facebook, websites, and mobile” (Adobe)

In the current, highly competitive world it is highly recommended that companies use tools similar to Brandwatch and Adobe Social to get a better grip on their social media campaigning. The case of ESPN should serve as an example of the potential of using social media in a successful way. Some of the features that these tools offer go beyond making a sale; they offer a chance to form a trusting relationship between the company and the customer. Although these tools offer a lot of ways to interact with the consumer and get their attention, they lack suggestions on how to capitalize on this attention. An approach forward should include ways in which this tool can be successfully applied to particular cases and how to translate the increased outreach into sales.

HYPOTHESIS

The purpose of this research is to answer the question of whether or not Twitter can be a viable source of new social business engagement opportunities. Tools such as Brandwatch and Adobe Social are designed to help companies connect with their clients and make meaningful connections with them. For the purpose of this research, I propose the following hypothesis:

If we collect previous examples of Tweets that represent opportunities for new social business engagement, we can train a model to recognize new tweets with the same potential. We can apply this model to real-time Twitter streams to produce a real-time list of social business engagements.

Engagement is bound to increase brand awareness which will help CSC’s sales in the long run. The algorithm and its ability to filter tweets can translate in increased revenue for CSC. The algorithm will take in a real-time stream of data from Twitter and filters the tweets based on content like hashtags and text. The filtered tweets will be sent to and end-user platform like a mobile app or dashboard. Users will be able to save and reply to tweets. The algorithm will use approximate string matching to find tweets similar to keywords and present opportunities for CSC to promote their content.

THE MODEL

Approximate string matching also known as fuzzy matching is the technique of finding strings that match a pattern approximately (rather than exactly). Fuzzy matching is a method that provides an improved ability to process word-based matching queries to find matching phrases or sentences from a database. It is extensively used in matching of nucleotide sequences, spell checking and in spam filtering. There are many different algorithms that can be used for approximate string matching such as the Damerau-Levenshtein, Jaro-Winkler and the Jaccard index.

Using the different available methods an algorithm was designed which would determine whether the 2 strings were a close match or not. The algorithm starts by calculating the distance between the 2 strings using different methods available in the ‘stringdist’ package of R. The methods used to calculate the different distances are Jaro-Winkler distance, Jaccard distance and cosine similarity. The algorithm uses a stepped approach to perform a series of checks on the two strings to ensure they are in close proximity to each other. Initially the program tokenizes each of the words of the strings.

```
# Pull data query from table and split into tokens

training <- queries_matrix[j,2]

trainingset <- strsplit(tolower(training), " ") # Tokenize training data

# Assign live data as data pulled from twitter and split into tokens

live <- tweets_matrix[,1]

liveset <- strsplit(tolower(live), " ") # Tokenize live data
```

By tokenizing the words of the string (sentence) it ensures greater accuracy, as the order of the string does not affect the distance. It begins by calculating the distance between two inputs strings which can be accessed using the stringdist package. The output is converted to the percentage of similarity between the 2 strings. Using the percentage of similarity calculated by the different methods, the algorithm compares them to preset threshold values and sorts them accordingly.

```
if (cosine_percent >= cosine_threshold){

  if (jaccard_percent >= jaccard_threshold){

    if(jw_percent >= jw_threshold){
```

As seen in the code snippet above the program compares the calculated percentage to a fixed threshold and if it's above the threshold, moves onto the next and more stringent test. If the strings pass all the tests, then it is considered a successful match.

The first test conducted was the cosine distance which gives the angular cosine distance between 2 inputs. The percentage of similarity between strings 'blue' and 'black' is 44.7%, this is because the frequency vector of the matching characters are just 1 in each of the strings. The formula used to calculate the distance is:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

As can be seen from the formula above the similarity is the dot product of the vectors divided by the product of the magnitudes of the two vectors.

The cosine similarity between the string 'blue' and 'black' can be found in the following way:

To make the string comparison easy we take the union of all the characters in both the words 'blue' and 'black' which will be {b,l,u,e,a,c,k}

Using the union of the vectors, the 2 strings can be arranged into frequency of occurrence vectors since they more accurately measure similarity. The frequency of occurrence measures the number of times a character in the union vector appears in each of the strings. The frequency of occurrence for blue will be the following:

a = {1,1,1,1,0,0,0}

The frequency of occurrence vector for black will be:

b = {1,1,0,0,1,1,1}

Using the vectors the cosine similarity can be calculated in the following way:

$$similarity = \frac{a.b}{\| a \| \| b \|} = \frac{2}{\sqrt{4} * \sqrt{5}} = 0.447$$

The percentage of similarity using cosine similarity between 'blue' and 'black' is 44.7%

In the stringdist package, the cosine formula returns a value between 0 and 1 which is the distance between the 2 strings.

```
cosine_dist <- stringdist(liveset,trainingset, method="cosine") # Cosine

cosine_percent <- 100 * (1- cosine_dist)
```

As can be seen in the code snippet above the distance is subtracted by 1 as to get the similarity between the 2 strings, and then it is converted to a percentage. The cosine distance showed high similarity rates to almost all the test strings and was assigned a high threshold. The cosine distance however acted as a good opening filter.

The Jaccard index, also known as the Jaccard similarity coefficient is a statistic used for comparing the similarity and diversity of sample sets. The percentage of similarity between the strings black and blue is 28.6% this is because the size of the set containing all the common characters from both the strings is much smaller compared to the size of the set containing all the different characters in the two strings. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

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

In the case of comparing 'black' and 'blue' each is defined to a set A or B. A intersects B (A ∩ B) 2 times with both words containing the letters 'b' and 'l'. A union B (A U B) is the distinct set of letters that are used to make up the words black and blue, these are b,l,u,e,a,c,k which are 7 distinct letters.

The Jaccard similarity between testing the words 'black' and 'blue' can be found in the following way:

$$J(A,B) = \frac{A \cap B}{A \cup B} = \frac{2}{7} = 0.2857$$

The percentage of similarity between black and blue using Jaccard index is 28.57%

Using the tokenized version of the input strings allows for greater accuracy. In the stringdist package of R, the jaccard distance similar to the cosine distance returns a value between 0 and 1.

```
jaccard_dist <- stringdist(liveset,trainingset, method="jaccard") # Jaccard

jaccard_percent <- 100 * (1 - jaccard_dist)
```

As can be seen in the code snippet above the distance is subtracted by 1 as to get the similarity between the 2 strings, and then it is converted to a percentage. This algorithm was picked over the other because it showed high similarity rates to approximately close strings and had a more vigorous test of false matches compared to the cosine distance.

Jaro-Winkler distance is a measure of similarity between two strings. The Jaro measure is the weighted sum of percentage of matched characters from each file and transposed characters. The Jaro-Winkler percentage of similarity between the strings blue and black is 70.7% this is due to the words having 0 transposed characters and having only 2 matching in-sequence characters. The Jaro distance between 2 given strings s1 and s2 using the following piecewise function:

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases}$$

- m is the number of matching and in-sequence characters for both the strings.
- t is half the number of matching and not in-sequence characters for both strings divided by 2.

Winkler increased this measure for matching initial characters whose weights rely on the type of string. Winkler introduced a prefix scale which gives more favorable ratings to strings that match from the beginning for a set prefix length . The Jaro-Winkler formula is:

$$d_w = d_j + (\ell p(1 - d_j))$$

- d_j is the Jaro distance
- ℓ is the length of the common prefix up to a maximum of 4 characters
- p is a constant scaling factor, set to 0.1

The Jaro-Winkler distance between ‘blue’ and ‘black’ can be found in the following way:

m = 2 as ‘bl’ are both matching and in-sequence

s1 = 4 and s2 = 5

t = 0 all matching characters are in-sequence so no transpositions are required

$$d_j = \frac{1}{3}\left(\frac{2}{4} + \frac{2}{5} + \frac{2-0}{2}\right) = 0.63$$

$$d_w = 0.633 + \big(2 * 0.1(1 - 0.63)\big) = 0.707$$

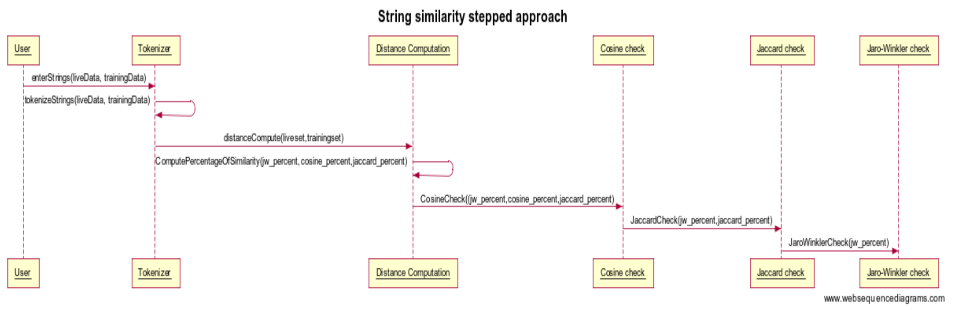
The percentage of similarity using Jaro-Winkler between ‘blue’ and ‘black’ is 70.7%

Similar to the use of previous methods the Jaro-Winkler distance in the stringdist package returns a value between 0 and 1.

```
jw_dist <- stringdist(liveset,trainingset,p=0.1, method="jw") # Jaro-Winkler

jw_percent <- 100 * (1 - jw_dist) # JW as percentage
```

The distance is subtracted by 1 to get the similarity and then multiplied by 100 to get the percentage of similarity. The Jaro-Winkler was selected as the final test because of the accuracy of the method. The Jaro-Winkler distances were seen over a wide array of values, showing very low percentage of similarity to false matches and relatively high to similar strings.


String similarity stepped approach
User
Tokenizer
Distance Computation
Cosine check
Jaccard check
Jaro-Winkler check
enterStrings(liveData, trainingData)
tokenizeStrings(liveData, trainingData)
distanceCompute(liveset,trainingset)
ComputePercentageOfSimilarity(jw_percent,cosine_percent,jaccard_percent)
CosineCheck((jw_percent,cosine_percent,jaccard_percent)
JaccardCheck(jw_percent,jaccard_percent)
JaroWinklerCheck(jw_percent)
User
Tokenizer
Distance Computation
Cosine check
Jaccard check
Jaro-Winkler check
www.websequencediagrams.com

The above sequence diagram shows how the R script runs and determines whether 2 entered strings are close matches or not. It is a condensed way to look at the algorithm.

Training Data	Live Data	Expected Results	Actual Results
"Cosmo Kramer"	"Cosmo Kramer"	Pass	Passed
"Cosmo Kramer"	"Kramer Cosmo"	Pass	Passed
"Cosmo Kramer"	"Comso Kramer"	Pass	Passed
"Cosmo Kramer"	"Csmo Kramer"	Pass	Passed
"Cosmo Kramer"	"Cosmo X. Kramer"	Pass	Passed
"Cosmo Kramer"	"Kramer, Cosmo"	Pass	Passed
"Cosmo Kramer"	"Jerry Seinfeld"	Fail	Failed (Cosine)
"Cosmo Kramer"	"Elaine Benes"	Fail	Failed (Cosine)
"Cosmo Kramer"	"George Costanza"	Fail	Failed (Jaccard)
"Cosmo Kramer"	"Kosmo Karme"	Pass	Passed
"Cosmo Kramer"	"remarK omsoC"	Pass	Passed
"Cosmo Kramer"	"Mr. Kramer"	Pass	Passed

"Cosmo Kramer"	"Sir Cosmo Kramer"	Pass	Passed
"Cosmo Kramer"	"C.o.s.m.o. K.r.a.m.e.r"	Pass	Failed (Cosine)

The table above visualizes the training and live data used along with the expected and actual results of the experiment. The algorithm was designed keeping in mind the types of data that are required and was tested using certain phrases to test the accuracy of it. The algorithm had good results catching the 3 fail cases and no false positives were found. However 1 false negative was thrown by the algorithm. Initially the algorithm had a fourth test which used the Damerau-Levenshtein method, this approach threw a lot of false negatives. The next approach of the algorithm was adopted to exclude Damerau-Levenshtein which reduced the false negatives to just 1. A major drawback of the algorithm is the threshold values as they will need to be updated every time different types of strings are being filtered. The updating of threshold values can be quite a tedious process as when the type of data is changed then the algorithm has to be rerun and a threshold value has to be selectively picked. It can become a major hindrance in the long run if the type of data frequently changes.

The next step was combining the algorithm with a stream of tweets and publishing the filtered tweets in an RSS feed. This allows for the a stream of tweets into R, followed by them being sorted in R using the approximate string matching algorithm and uploaded to an MySQL database. There are several R packages used which are TwitteR, RMySQL and stringdist package. To access the Twitter server a Twitter application must be created which can be done using the Twitter website. Authentication to Twitter is done using OAuth which is an open standard to authorization. The OAuth provides secret tokens to access the protected resources hosted by the resource server (Twitter). The authenticating credentials are stored in a file and the file is loaded while the script is executed, which allows for a swift connection to the server. A connection is established to FutureTense MySQL database using RMySQL, which allows SQL queries to be executed from within R.

```
load("twitteR_credentials") #load twitter credentials

registerTwitterOAuth(twitCred) #Check twitter connection status

#connect to the database

ch <- dbConnect(MySQL(), host = "cscfuturetense-db-new.crmcnjxc1gst.us-west-2.rds.amazonaws.com", user= "cscfuturetense", password = "*************",client.flag=CLIENT_MULTI_STATEMENTS)
```

Two tables were created on the FutureTense MySQL database, each table was meant to contain certain information that would be used during different process. The first table is the queries table which contained 2 columns, the query column where the question or key phrase which would be used to filter the tweets would be stored and a category column which contained the part of the company or the industry it corresponds to. The program pulls the category from the SQL table and searches Twitter for tweets containing the category as a keyword. Once the tweets the pulled from the Twitter server they are put into a data frame called refined_tweets_df in R. The data frame is cleaned and only the important columns are taken into account.

These are text (the tweet), screen name (Twitter handle), id (unique tweet id), created (date and time of creation) and the retweet count (the number of times it has been retweeted). There are certain drawbacks of using the Twitter package to pull the tweets from the Twitter server. The package and in some cases R is not fully equipped to handle all the different types of objects it gets from the Twitter server. When these objects are encountered an error is thrown which disrupts the program.

```
#search from twitter and convert into a dataframe

twitter_search <- searchTwitter(twitter_searchword, n =
1000,cainfo="cacert.pem")

tweets_df = twListToDF(twitter_search)

#clean up data from twitter and convert into a matrix

refined_tweets_df <-
tweets_df[,c("text","screenName","retweetCount","id","created")]
```

The data frame is converted into a matrix as to enable the easier data access and called tweets_matrix. The text column of the tweets_matrix is inputted into the string matching algorithm as the live data. The training data is obtained from the queries table on MySQL server from the query column. Using the algorithm, the matching percentages for the Jaro-Winkler, cosine and Jaccard distances are calculated. A data frame containing all the required information is created and the tweets are filtered out using the pre-determined threshold values.

```
# convert into a matrix

tweets_matrix <- as.matrix(refined_tweets_df)

#Pull data query from table and split into tokens

training <- queries_matrix[1,2]

#update dataframe

dataframe_1 <-
data.frame(refined_tweets_df,cosine_percent,jaccard_percent,jw_percent)
```

The threshold values were determined through experimentation of a set of tweets where the matching percentage of each tweet was examined and the threshold values were picked based on the usefulness of each tweet. The threshold values currently set in the prototype are:

```
cosine_threshold <- 94

jaccard_threshold <- 63

jw_thershold <- 70
```

The data frame is further filtered with the removal of elements under the threshold values. The resulting collection is a set of tweets with utility to CSC. These tweets are pushed back on the MySQL database to a second table where they are recorded. The final part of the program is to push the tweets into an RSS feed. Several RSS feeds can be formed and can be classified based on the category from the queries table. The

RSS feeds contain conversation that individuals within CSC can use to jump into conversations and drive content.

To create an RSS feed of the tweets an Amazon EC2 Ubuntu instance was launched. On the instance LAMP stack was installed. LAMP is an acronym for an archetypal model of web service solution stacks, originally consisting of largely interchangeable components: Linux, the Apache HTTP Server, the MySQL relational database management system, and the PHP programming language. As a solution stack, LAMP is suitable for building dynamic web sites and web applications (Unknown). Using the LAMP stack, a PHP script was written which pulled data from the MySQL database and converted it into a XML format which is followed by all RSS. This is done by pulling all the tweets from the MySQL server into the instance and then formatting them according to the XML format.

```
// The XML structure

$data = '<?xml version="1.0" encoding="UTF-8" ?>';

$data .= '<rss version="2.0">';

$data .= '<channel>';

$data .= '<title>CSC : Twitter Monitor </title>';

$data .= '<link>https://c3.csc.com/groups/futuretense/projects/can-twitter-be-a-legitimate-intelligence-tool</link>';

$data .= '<description>Twitter data mining</description>';

foreach ($rs_post as $row) {

    $data .= '<item>';

    $data .= '<title>'.$row['tweet'].'</title>';

    $data .= '<link>'.$row['url'].'</link>';

    $data .= '<description>'.$row['handle'].'</description>';

    $data .= '<category>'.$row['Category'].'</category>';

    $data .= '</item>';

}
```

With the PHP script running whenever new data is added to the MySQL tables, that item will automatically be available in the RSS feed. The RSS feed can be accessed via http://ec2-54-148-157-118.us-west-2.compute.amazonaws.com/rss.php.

CONCLUSION

The hypothesis was really investigating whether a model can be trained to find competitive intelligence from social media. The initial objective was to find blogs or posts that had business potential and value to CSC. These posts could range from simple topics such as effective communication to more pertinent issues like threats to cybersecurity or exploiting big data to predict real world outcomes. Because Twitter is one of the most active social networking sites at the moment, with more than 280 million users logging onto the website monthly and 500+ million tweets being posted daily, using such a network to monitor was a logical choice. Initially people were quite skeptical of Twitter's ability to deliver any sort of information, often citing its restriction to 140 characters as too few to get a message across or believing that it is full of people advertising their product or themselves. That notion is quickly changing with the interest in Twitter analytics.

The next step was to find useful tweets from Twitter and train the model to recognize tweets with similar potential. With the use of R and several of its packages this was successful. Visualize being at a party and being able to listen to all conversation happening around a topic you are well-informed while also having the ability to jump into the conversation and offering your opinion; this is the ability the model affords CSC. The model is being run hourly to find the tweets that will help CSC drive content and increase its reputation.

This model is a part of a bigger vision, a vision to establish a social command center within CSC. A command center is usually an entity associated with air travel. The command center is responsible for tracking a flight from take-off to its landing. A social command center also known as a social listening center works in a similar way, its objective is to monitor the company's presence on social media. A command center usually consists of a large screens and a group of dedicated individuals that monitor the social pulse of a brand. A social listening center brings the company with many different ways to tackle social media such as customer service and public relations, using the command center the team can easily visualize and monitor the type of complaints the company is receiving. The social command center also allows for regional benchmarking, by keeping track of metrics in certain areas the across various locations or regions, marketing, sales, and product teams can identify how the business's presence varies throughout the world and why that is the case. Additionally, for businesses looking to expand to new territories, being able to identify and engage with influencers or advocates in target locations is crucial.

The competitive intelligence model is one of such tools that should be employed by CSC to incorporate into and create a social command center. Combining the model and a social command center, CSC can expect to get engageable opportunities. That may mean identifying conversations from important influencers and having the social team craft responses together. The big-screen collaborative environment around social engagement promotes a much more unified voice in a company's social outreach. The model combined with the social listening center can also real-time marketing prospects. Companies no longer have to wait to understand how the public is receiving their campaign – now they can keep track of the response as it unfolds online. Considering how quickly online communities can develop strong opinions, staying aware of those views and being agile to the rapidly shifting online sentiment can amplify a campaign's success or mitigate its failure. A social command center will also be successful in increasing brand online presence. Using the command center CSC can monitor what is being said about their clients, what is happening in their markets and industries, what's being said about them online. Using this data CSC can help them foresee problems and try to fill their unexpressed needs.

Companies such as Gatorade, Walmart and Jaguar have successfully implemented social command centers. They have been praised for their efforts to pioneer this technology and setting the foundation of digital leadership. Companies such as Amazon and Salesforce are known to have them at their conferences so conference attendees can have a better grip on the effect the conference is having around the world. The CEO of Target, Brain Cornell, ditched his “newly redone CEO’s corner suite on the 26th floor” and moved to a “smaller office down the hall that is only steps away from the company’s global data nerve center” to be closer to the social command center, allowing him to have a deeper view of his business. He plays an active role in Target’s social media presence, “dropping in every morning and insisting on two updates a day.”

The next step would the construction of a social command center. Being a company as large as CSC with offices in more than 60+ countries more than one command center would be required to track the progress all over the world. A command center could be setup in every major continent. Having command centers setup across various regions will give CSC a chance to monitor the whole world during the peak of the day. Each command center could pass the next the responsibility as the time zones progress. One of the command centers would be setup in CSC headquarters in Falls Church, Virginia. Having a command center setup in the company’s headquarters would give CSC executives opportunities similar to that of the Target CEO. Having access to a command center just a few steps away will aid in their decision making and more thorough conflict resolution.

ACKNOWLEDGEMENTS

I would like to thank Kyle Zellman, Data Scientist, CSC's ResearchNetwork (@khzellman) for consistent help throughout the project and the paper. For providing numerous revisions and edits and for mentoring me during my time at CSC.

I would also like to thank Jerry Overton, Senior Principal, Data Scientist. CSC's ResearchNetwork and CSC Distinguished Engineer (@JerryAOverton), Steven Patrick Melanson, Data Scientist, CSC's ResearchNetwork (@DataScience_SPM) and Theyaa Matti, Data Scientist, CSC's ResearchNetwork (@TheyaaMatti) for their insights during ideation and their enthusiasm in helping me whenever I hit a roadblock.

I would also like to thank Nigel Barron, Research Analyst, CSC's ResearchNetwork (@NigelBarron) for bringing constant attention towards my project and sharing his vision for command centers and their implementation within CSC.

I would also like to thank Sherri-Hartlen Neely, Senior Principal, CSC's ResearchNetwork (@shartlen) for managing the business side of my project and being one of the early adopters.

REFERENCES

(n.d.). Retrieved October 2014, from Brandwatch: http://www.brandwatch.com/how-it-works/#

(n.d.). Retrieved October 2014, from Brandwatch: http://www.brandwatch.com/wp-content/uploads/2014/09/FINAL-ESPN.pdf

Adobe. (n.d.). Retrieved October 2014, from http://success.adobe.com/assets/en/downloads/product-overviews/24408_adobe_social_solution_overview_ue_v2.pdf

Fidelman, M. (n.d.). Retrieved October 2014, from Forbes: http://www.forbes.com/sites/markfidelman/2013/04/25/10-lessons-from-the-top-25-most-engaged-brands-on-twitter/

Keefe, K. O. (n.d.). Retrieved October 2014, from http://kevin.lexblog.com/2012/05/30/twitter-and-linkedin-ready-sources-of-competitive-intelligence-for-law-firms/

Martin, T. (n.d.). The Invisible Sale. Retrieved from http://tommartin.typepad.com/positive_disruption/2012/02/twitter-marketing-tactics.html

Unknown. (n.d.). Retrieved from http://en.wikipedia.org/wiki/LAMP_%28software_bundle%29

Yueng, K. (n.d.). The Next Web. Retrieved October 2014, from http://thenextweb.com/insider/2013/07/18/adobe-social-update-brings-predictive-publishing-and-integration-with-foursquare-instagram-and-linkedin/

###