Weather Tracker

Using Social Media to Track Weather Sentiment.

Bryce Pedroza
CIDSE
Arizona State University
Tempe, USA
bapedroz@asu.edu

Ariane Middel
School of Arts, Media and Engineering
Arizona State University
Tempe, USA
Ariane.Middel@asu.edu

I. Introduction

Thermal comfort can be difficult to define. The ideal thermal comfort ranges for each person. However, it is accepted that it is the experience of ideal conditions for humans' physical and mental health. Being in such conditions is known to impact an individual physically, physiologically, and psychologically positively [1].

Because of this, it is important to understand what environments allow people to experience thermal comfort. Factors such as clothing and metabolic rate are known to have a factor in an individual's thermal comfort [2], but these can not be generalized for everyone. Instead, research has shifted to study external environments and their impact on thermal comfort. Researchers have identified a few key environmental factors that are known to have an impact on an individual's thermal comfort: heat index, humidity, and temperature [3].

Gathering information on these environmental data points would likely prove effective at identifying ideal weather conditions for experiencing thermal comfort. Presently, this information is captured via surveys or questionnaires. However, people are not obligated nor required to fill out and return these forms. These shortcomings can lead to homogeneity or bias in the results. Therefore, not only are the results limited, but the people who return the results are typically not representative of overall human thermal comfort levels of the United States [3].

To address this problem, we turn to social media and specifically Twitter, which has more than 152 million daily active users as of early 2020 [4]. More importantly, 80% of all Twitter users primarily use the mobile application, which allows them to post while going about their day. Users have the ability to geotag their tweets, which associates the tweet with a given location. Therefore, with these conditions, Twitter is a promising social media platform to capture users' posts and relate them to weather-related data to identify conditions that exhibit high levels of thermal comfort.

II. State of the Art

Prior research has utilized Twitter for sentiment analysis in a multitude of different concentrations of research. Twitter has been leveraged for gauging public support of politicians and policies [5], assessing public health [6], or monitoring evacuation strategies during natural disasters [7]. Furthermore, researchers have done work in assessing a population's perspective on different weather conditions, such as temperature, humidity, wind speed, UV indexing, etc. [8]. However, these weather collection applications do not work in real-time. The system proposed by Giuffrida et al. calculates an individual's assessment of weather conditions after the tweet has already been collected, not allowing for real-time analysis of data as it is being streamed [3]. The collection of the tweets was also not performed in real-time, adding more latency to the entire system. Requiring additional time and computation does not allow for these systems to be widely adopted. Full-scale urban climate applications allow results from prior works to become more widely adopted and is an important step to drive urban climate decisions at a local, state, national, or global level [6].

III. Methodology

The proposed application draws on the inspiration of previous research along with personal knowledge to build a full-scale application to identify thermal comfort levels across the United States. This information should prove beneficial in gauging ideal thermal comfort levels in different regions to assist in the understanding of what weather conditions have the highest impact on thermal comfort.

The primary goals of the application are as follows. First, the application should identify if a geotagged tweet contains information about current weather conditions. Once determined, the sentiment of that tweet should be recorded to determine if the tweeter is mentioning the weather positively or negatively. Furthermore, the geotagged location of the tweet is to be extracted and weather conditions at the time the tweet was shared should also be collected. The tweet metadata associated with the tweet and the supplemental data is all collected into a datastore that can be extracted via a RESTful API. A front-end application thus requests the information from the API and displays the tweet information and overall trends in a meaningful way and allows for data exploration to observe underlying trends in the data. These components are to be developed and deployed within a cloud environment to allow for anyone to easily access the application and underlying data.

The building of this application was broken into 6 key components. Each component had its own challenges and

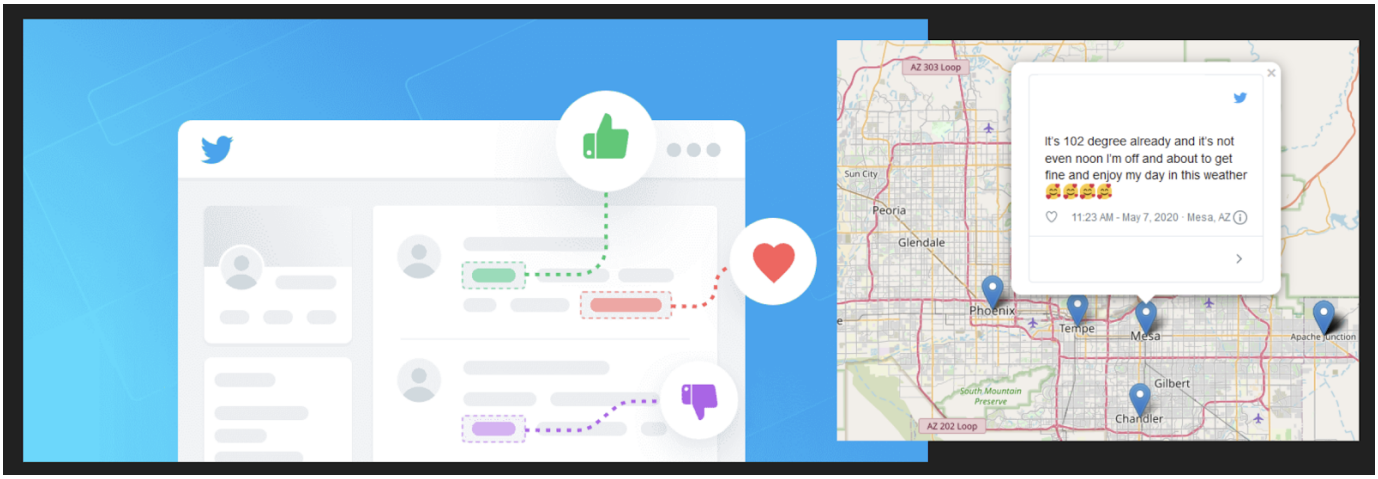

Fig. 1. Proposed design to automatically detect geotagged tweets related to weather and capturing sentiment.

techniques that made each part of the project an enjoyable experience along the way.

A. *Tweet Preprocessing*

A Twitter API token was provisioned to acquire access to their tweet streaming platform. When streaming tweets with Twitter's standard API, only one query parameter is allowed for filtering tweets. The most important aspect of this project is to tie a tweet to the location from where it was tweeted. Therefore, the filter chosen for this was a geolocated bounded region of the United States and parts of Canada and Mexico. The bounded region was set to the following: Point 1: -126.210938, 24.686952 Point 2: -66.708984, 50.457504 Point one served as the upper left of the bounded region, and a square was built connecting to point two. Once the stream begins, all geotagged tweets from that location are captured in real-time.

However, only weather-related tweets should be captured. Therefore, additional logic had to be implemented to filter out any extraneous tweets. Giuffrida et al. proposed a list of keywords that can be used for gathering tweets related to the weather [3]. Below is a list of the keywords chosen to filter tweets. To determine if a tweet had a corresponding keyword, a function was built to vectorize the tweet, such that its contents were converted into lower case and inserted into a list, and then compared against all the keywords to determine if any existed in the tweet.

```
KEYWORDS = [
"arid", "autumn", "blizzard", "blustery", "breeze", "chill",
"chilly", "cloudy", "cold", "colder", "coldest", "downpour",
"drizzling", "dry", "flurries", "fog", "foggy", "freeze",
"freezing", "frost", "gale", "hail", "haze","heat", "hot",
"hottest", "humid", "humidity", "mist", "muggy", "overcast",
"rain", "raining", "rainy", "sizzler", "sleet", "snow",
"snowing", "snowy", "springtime", "storm", "sun", "sunburn",
"sunlight", "sunny", "sunscreen", "sunshine", "sweltering",
"temperature", "thunder", "umbrella", "warm", "warmer",
"warmest", "weather", "wind", "windy", "°C", "°F"
]
```

Upon streaming the tweets and capturing ones that contained these keywords, there were false positives that appeared. Some of these tweets, while tweeting about the weather, did not contain any sentiment. These were often the result of twitter bots written by other developers, or weather stations or channels tweeting current weather conditions. Other false positives contained some of the keywords but were only mentioned as part of a larger phrase not relating to weather. Therefore, additional logic had to be put in place to ensure that the tweets did not contain any of these words or phrases typically associated with false positives. Below is a list of the words or phrases to ignore. Most of this list mirrors Giuffrida et al, however, I also added some other keywords and phrases that often created false positives.

```
TOIGNORE = [
"wind:calm", "humidity up", "humidity down", "temperature
up", "temperature down", "dew point", "today's records",
"trump", "good morning", "drinking", "gusting", "today's
forecast", "barometer", "weather now", "hiring", "can you
recommend anyone for this job", "diabetic", "just posted a
photo@", "ice cold","@realDonaldTrump", "POTUS", "hot take",
"hot dog", "corona", "coronavirus", "cold brew", "cold beer"
]
```

Once the content of a tweet passed both of these checks, it was then considered valid for classification. To save a tweet for classification, the content of the tweet, and any additional relevant information was recorded to a CSV file. The data captured included the tweet-id, tweet content, user-id who posted the tweet, and the time at which the tweet was posted. The stream ran over the course of April 7th, 2020 - April 10th, 2020. In total, 11304 tweets were captured during this time period.

B. Tweet Weather Identification Training

Of the 11304 tweets captured over the course of four days, a total of 1890 were used to build the classification model. Roughly 500 were selected across each day the data was recorded. The 500 tweets were randomly selected to ensure a fair representation of the data was preserved. Each of the 1890 tweets was manually read and assigned a 1 if the tweet contained a subjective opinion on the weather, and a 0 otherwise.

Once the training dataset was classified, the next step was to build the model that would predict if a tweet mentioned the weather. A logistic regression model was chosen as the primary driver for the binary classification of the data. Logistic regression on tweets has been used in prior research as a way to binary classify data and proved promising as a simple approach that does not introduce much more additional overhead to the system [9]. Logistic regression was chosen rather than linear regression due to the bounds that the data must lie in. The results captured by the logistic regression are discrete and bounded between 0 and 1. Linear regression, however, is a continuous modeling technique that allows for it to model probabilities less than 0 or greater than 1 which are counter-intuitive. Also, it allows for a slightly more complex data fitting rater than a simple linear function produced by linear regression.

Python's scikit-learn library was leveraged to implement the logistic regression model. The training data CSV was input and read into a Python Pandas data frame to allow for easier data manipulation. Each unique word in the set of training tweets was placed into a feature vector, and once produced, each tweet was transformed into its vector representation. Consider the following example below. The Logistic model was trained using a 75% training and 25% testing split. The accuracy of this model will be shared in the results section below.

```
"It is hot outside"
"I am outside and this rain is terrible"
{
  it: 0
  is: 1
  hot: 2
  outside: 3
  i: 4
  am: 5
  and: 6
  this: 7
  rain: 8
  terrible: 9
}
[1, 1, 1, 1, 0, 0, 0, 0, 0]
[0, 1, 0, 1, 1, 1, 1, 1, 1, 1]
```

Here you can see a quick example of vectorization using only two tweets.

C. Tweet Sentiment Analysis Training

Python's Natural Language Tool Kit (NLTK) was used extensively to train the sentiment analysis model. A Naive Bayes classification model was trained to predict if a tweet was positive, represented as a 1, or negative, represented at a 0. [10]. The training set was a list of positive and negative tweets available via NLTK. The dataset of tweets contained 5000 positive sentiment tweets and 5000 negative sentiment tweets. The noise from the tweets, such as punctuation and URLs, were removed from the tweets. Punctuation meant to convey emotion, such as :) or :(, were maintained in the tweets. The vectorization approach used to build the weather detection algorithm was also utilized to transform tweets into a feature space that the Naive Bayes classification model could understand. The model was trained using 70% of the tweets for training and 30% for testing.

Naive Bayes classification is a simple machine learning classification algorithm that produces a probabilistic classification system. To determine which label to apply, it applies a Maximum A Posteriori (MAP) decision rule. This density-based decision model applies labels according to where the data lies in accordance with the distribution of the training data. [11]. The result is a classification system that takes a tweet as an input and predicts the sentiment for that tweet as either positive or negative. The accuracy of the model will be shared in the results section below.

D. Real-Time Data Collection

The streaming setup utilized for the collection of training data was also used for the main data collection. Tweets were filtered out using the same geotagged bounded region, keywords, and phrases to ignore. Once a tweet satisfied those conditions, it ran through the weather classifier to determine if the tweet was actually subjectively mentioning the weather. If the classifier determined it was not, then the tweet was ignored and the service continued. Otherwise, additional data collection was performed to augment the weather-related tweet.

The first step was to record the sentiment of the tweet using the classifier built in the prior section. The next step was to collect the weather data where the tweet originated. To do this, a full name related to the originating location of the tweet was included in the JSON object returned from Twitter's API. This location's full name was passed into a geocoding python library that calculated the latitude and longitude of the tweet. The current weather at that latitude and longitude were recorded with the Darksky API. The API contains valuable information, such as the summary of the current conditions, temperature, humidity, wind speed, cloud coverage, UV index, etc. All of this information along with the metadata within the tweet itself was then sent to an external Azure Cosmos Database. Cosmos holds information as key-value pairs, which meant sending the Python dictionary was a fairly straightforward task.

E. Data Retrieval

An API was built using Python's Fast API library. The API serves as a secure way to communicate with the data within the Cosmos database. The API has three RESTful endpoints. The first takes a single latitude and longitude and retrieves all tweets within a mile radius around the given location. The radius has the ability to be adjusted to allow for a larger search space. This endpoint should be beneficial in allowing a user to query the database for all tweets originating from a city and some neighboring cities.

The next API endpoint grabs all recent tweets for a given bounded region in terms of latitude and longitude. Much like how Twitter allows users to query a region, these two points serve as the top left and bottom right of the bounded region. The region is set to the same region from which the data was collected by default. However, to limit latency only tweets recorded in the last 12 hours are returned to the user. This time frame cannot be adjusted. This endpoint serves as a way to display all recent tweets in a given region, and it does not represent the entire dataset.

The endpoint that represents the entire dataset is the last endpoint. This endpoint grabs all tweets in the Cosmos collection that houses all the recorded tweets. As the number of tweets grows, the time it takes to return all the data increases, so it is not particularly optimal for applications that require speed. However, it can be used to perform advanced analysis of all the data collected.

F. Data Display

A React JS application was created to display the results from the data collection. To do so, it queries the API and creates two views. The first view is the map view. The map view collects all tweets recorded in the last 12 hours via the API and displays them on the map according to where they originated. Clicking on the pin allows for a full view of the tweet. This view allows for data exploration and to observe trends that could not be seen otherwise. The second view contains all the statistics related to the project. Using the get all endpoint, this view aggregates and bins all of the data to determine underlying trends across the entire dataset. Specifically, it highlights public sentiment at different temperatures, humidity, UV exposure, and general conditions.

G. Code Maintainability and Testability

This code uses multiple different package managers to create consistent development environments. For Python, Pipenv is used as the package manager, and Yarn is the package manager for the React application. Both solutions allow for libraries to be pinned to specific stable versions, therefore running the code on another machine is simple. For production builds, Both the API and application are put inside docker containers with an NGINX container serving as a reverse proxy service used to manage communication in out and of the other two containers. This allows for multiple applications to be accessible by exposing only a single port. The three containers are managed via docker-compose, which manages orchestration for multiple Docker containers with a single configuration file.

H. Application Deployment

In order for other researchers to use this application, it was deployed in the cloud using Azure's PaaS solution. The Docker images are hosted in a private container registry in Azure that can be referenced for production builds. The Azure PaaS instance provisioned takes the docker-compose file mentioned prior and automatically pulls the images down from the container registry, builds them, and deploys the application over HTTPS. The codebase exists in GitHub and executes a CI/CD workflow whenever the master branch is updated. This workflow automatically builds, tags, and pushes the images to Azure automatically. The App Service in Azure then sees that the images in the container registry have been updated, and the app automatically rebuilds and redeploys without any manual intervention. The application can be found here: https://weathertracker.azurewebsites.net

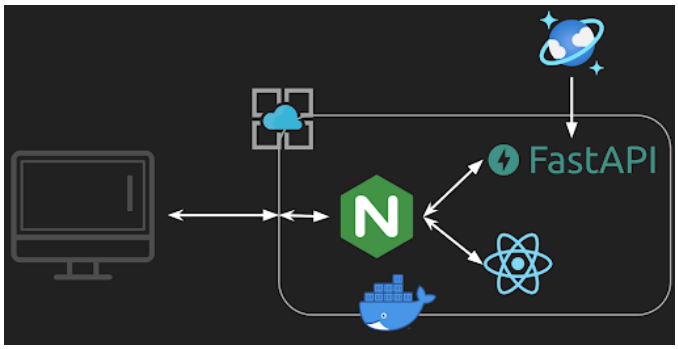

Fig. 2. Full architecture overview.

IV. Results and Discussion

A. Weather Training

Partitioning 75% of the 1890 classified tweets to training and 25% to testing gave an accuracy of 73.11% on average over the course of 10 different tests. These results were calculated by observing how the model changes utilizing different random seeds for randomizing and splitting the data. While these are not perfect results, they were more than good enough for the purpose of the project. Furthermore, the classification models tended to be more conservative in their classification of tweets related to weather which produced more false negatives than false positives.

B. Sentiment Training

The Naive Bayes model implemented on the classified tweets from NLTK using 70% for training and 30% for testing yielded a sentiment detection accuracy 99.12% on average over 10 different tests. While the model touted these results during classification, it certainly felt as though this was not the actually observed accuracy of the model, but it still proved extremely accurate at predicting the sentiment of most tweets. However, the real number was noticeably lower than 99%. This likely has to do with context, such as sarcasm, which is something we have discussed in quite some depth during class. Unfortunately, that is a tricky thing to detect, and I did not want this project to spiral into a machine learning classification project, so I ran with what I thought could get good results fast.

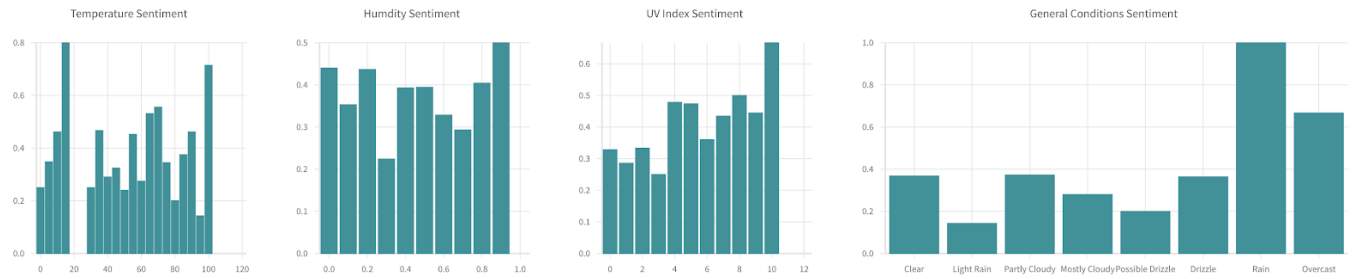

Fig. 3. Visualization highlighting the sentiment of different weather vectors available that are known to have a high correlation with thermal comfort.

C. Data Streaming

Over the course of 48 hours, 1203 tweets were captured in the established bounded region, which equates to roughly 25 tweets per hour. This number feels low, but let's consider why this may be the case. First, the code running the stream was being run locally and was prone to the socket establishing the connection to the Twitter API timing out unexpectedly. Unfortunately, the code written to stream this data was not written in the most fault-tolerant way, so while it would catch some exceptions and reestablish the socket, it would not do this in all cases. This caused the stream to not have 100% uptime throughout the entire 48 hours during which this data was captured. Thankfully, these errors were caught and patched every time a new one occurred, and the code became more fault-tolerant. In a more recent 12 hour stretch, the stream captured 452 tweets. Roughly 38 tweets per hour. Compared to the 48-hour snapshot, this 12 hours stretch saw a 41% increase in the velocity of tweet capture. And thankfully the stream is still running, so more and more tweets will be added over time.

D. Sentiment Analysis

The reports captured the overall trends across all the recorded tweets. Temperature sentiment shows two distinct peaks that trend upwards from left to right. The first one is a small trend in low temperatures at 0 degrees Fahrenheit up to 15 degrees Fahrenheit. The second peak is a gradual ascent upwards peaking at 100 degrees Fahrenheit. Here is where we can start observing that there might be regional trends in the data, as people are appearing to speak highly of two vastly different weather conditions. However, these results show aggregate information and there currently is not a filtering system in place for these graphics.

Humidity sentiment, at first glance, does not appear to feature any noticeable trends other than a small spike at higher levels of humidity. The one trend that is present is that no level of humidity reports a sentiment score higher than 50%. UV Index appears to have the most striking results of all the data points studied. There is a clear incline that presents itself as the UV index increases. This appears to reveal that people tend to be happier the sunnier it is outside regardless of location. Lastly, there is the sentiment across all overall weather conditions. It seems surprising that rainy conditions have the highest overall sentiment when considering the positive trend in the UV Index data, but this might be a cause of not having enough data points to capture the true sentiment around the data. More data points would have been captured to get a better understanding of these graphics.

V. Future Work and Conclusion

The results of this project definitely prove promising. The way in which the project was developed should allow for many other projects to be implemented in a similar way, hopefully addressing the current lack of urban climate applications that are built at scale. However, the project was not perfect, and there are certain areas that can be improved. First, in terms of the weather classification, more tweets could have been added to the classification model. Maybe this would have improved the model significantly, but if it did not, more research into different models could be done. For the tweet sentiment classification, additional data should have been included to specifically train on detecting sentiment in weather-related tweets. This would hopefully allow for better sentiment results. As mentioned earlier, the streaming service could have been made more fault-tolerant and deployed into the cloud for consistent streaming, but it was not needed for a final product.

The current methodology for querying the API and performing all the data wrangling in JavaScript on a client machine is not the best practice. The best solution would have been an endpoint to prepare the information and send it out to the client machine to render without requiring additional data manipulation. Finer grain filtering could be implemented to determine what weather conditions different regions prefer. Lastly, more graphics can be produced to allow researchers to study how people react to other weather conditions.

It would have been awesome to gather actual survey data and compare it to the results gathered from this project. If the results from surveys are similar to this data, then this form of data collection for identifying thermal comfort should certainly be considered. Overall, I am incredibly happy with this project and how I was able to apply the concepts I learned from this class into a full-scale urban climate application. This project is certainly not an end-all solution, but it should allow users to start identifying in what conditions people experience thermal comfort.

REFERENCES

[1] S. Shooshtarian, "Socio-economic factors for the perception of outdoor thermal environments: Towards climate-sensitive urban design," *Global Built Environment Review*, vol. 9, pp. 39–53, 10 2015.

[2] M. Nikolopoulou and K. Steemers, "Thermal comfort and psychological adaptation as a guide for designing urban spaces," *Energy and Buildings*, vol. 35, no. 1, p. 95–101, 2003.

[3] L. Giuffrida, H. Lokys, and O. Klemm, "Assessing the effect of weather on human outdoor perception using twitter," *International Journal of Biometeorology*, vol. 64, no. 2, p. 205–216, Jul 2018.

[4] Y. Lin, "10 twitter statistics every marketer should know in 2020 [infographic]," May 2020. [Online]. Available: https://www.oberlo.com/blog/twitter-statistics

[5] M. A. Bekafigo and A. Mcbride, "Who tweets about politics?" *Social Science Computer Review*, vol. 31, no. 5, p. 625–643, Jul 2013.

[6] M. Illieva, R.T., "Social-media data for urban sustainability," 10 2018, pp. 553–565.

[7] Y. Martín, Z. Li, and S. L. Cutter, "Leveraging twitter to gauge evacuation compliance: Spatiotemporal analysis of hurricane matthew," *Plos One*, vol. 12, no. 7, Jul 2017.

[8] T. Stathopoulos, H. Wu, and J. Zacharias, "Outdoor human comfort in an urban climate," *Building and Environment*, vol. 39, no. 3, p. 297–305, 2004.

[9] S. Indra, L. Wikarsa, and R. Turang, "Using logistic regression method to classify tweets into the selected topics," 10 2016, pp. 385–390.

[10] "Natural language toolkit." [Online]. Available: https://www.nltk.org/

[11] "Cs 109: Maximum a posteriori (map)," *Stanford University*.