EECS E6893 Big Data Analytics Project Progress Report
Search for a Connection: Energy Demand & Twitter Trending Topics

Kevin Murning
UNI:
kmm2344

Rifqi Luthfan
UNI:
rl3154

Rohan Raghuraman
UNI:
rr3417

Abstract

The goal of this project is to evaluate the hypothesis that spikes in energy demand for a given region can be reliably predicted by the monitoring and analysis of social media activity using big data analytics. Can Twitter activity formulated into topics be linked with a meaningful, causal relationship with energy demand spikes? The approach we take to evaluate this hypothesis is to explore the problem in the inverse. First, hourly energy demand data is collected for New York State for a 10 year period. A time series forecasting model is then used to control for seasonality and extract the dates of anomalous demand spikes. Twitter activity for the given region on the dates specified is then collected, and a topic modeling algorithm is used to determine the topics of interest during the period of the demand spike. Finally, using big data visualization techniques and power systems domain expertise we determine if the topics corresponding to demand spikes are merely correlations or a meaningful causal relationships, thus allowing us to form a hypothesis that twitter activity relating to certain topics of interest is likely to cause a spike in energy utilization.

1. Introduction

Effective energy demand forecasting plays a vital role in power systems. Sufficient resource allocation and economically viable pricing both rely on accurate forecasts of changes in energy demand. Some trends are easy to infer, such as an increase in demand in winter months due to use of heating. However, outlier events can cause spikes in demand that are hard to predict. For example, in the case of the COVID-19 pandemic, residential electricity demand increased due to the number of people staying at home. How can we predict anomalous changes in demand such that electricity utilities can adequately prepare for an event? Social media provides a useful data resource from which real-time social, economic and political information can be utilized to inform decision making through big data analyt-

ics. In the case of energy demand forecasting, social media activity may hold insights that could allow the prediction of anomalous spikes in energy demand. For example, it would be advantageous for energy utility companies to know that a set of trending topics on twitter were highly correlated with future spikes in energy demand, allowing them to prepare adequately. Our hypothesis is that there exists a collection of trending topics on twitter that meaningfully relate to increases in energy demand. This is an exploratory project with the goal of validating or invalidating this hypothesis. In order to do this, first we will collect 10 years of power utilization data for New York State. Next, we will use time series forecasting using this data to detect anomalous spikes in energy demand. Using the dates on which these spikes occurred, we will collect tweets occurring on those dates in the region of interest. The twitter data will be used to perform topic modeling, ultimately giving us a collection of topics that correlate with spikes in energy demand. Finally, using data visualization and the collected topics, we will use domain expertise in power systems to attempt to validate whether a spike in demand and a collection of topics have a causal relationship.

2. Related Work

2.1. Mining Twitter

Twitter provides an abundant resource of data that is open and easy to collect. Many previous works utilize twitter to collect data for time series forecasting. Using social media to conduct time series forecasting appears frequently in financial analysis literature, for example cryptocurrency price prediction [3] and crude oil price prediction [6]. In the case of energy demand forecasting, utilising social media as a scalable sensing solution for real time energy demand forecasting has been explored in [2] and [4]. In [2], the authors model social media networks as a network of sensors, where each user represents a noisy sensor. However, unlike conventional sensors which output numerical data, these social media sensors output textual data. Thus in order to apply statistical methods to the data in question, the data

needs to be pre-processed appropriately. The authors use topic modeling to cluster textual data and form a numerical representation with which to model. A different approach to energy demand forecasting using social media is taken in [4]. In this case, the authors do not use the content of social media activity, but rather meta-information such as the number of tweets posted to train a neural network to predict energy demand for a given region. While only utilizing meta-information is more simple, determining causal links between the content of tweets and real life events as the authors do in [2] appears to be a more powerful approach.

2.2. Smart Grids and Energy Demand Forecasting

Efficient and effective grid management plays an essential role in modern society. Economic growth is fundamentally dependent on successful management of electrical infrastructure [5]. One important aspect of grid management is energy demand forecasting, which ensures that electricity demands are met and costs of energy production are effectively managed. In recent years, energy utilities has been increasingly incorporating smart grid infrastructure. Smart grids serve as an enhancement to traditional grid infrastructure by utilizing the collection and processing of data from sensors to improve co-operation and communication in service of predicting and reducing spikes in energy demand [7]. For example, networked sensors could be placed in the electricity infrastructure in residential units, allowing for data collection of power demand changes at a granular scale. However, the authors in [2] highlight that this is both expensive and impractical. This motivates the use of social media as a data source for energy demand prediction. A comprehensive review of modeling techniques for demand forecasting is presented in [5]. The most commonly utilized models are Neural Networks (including convolutional neural networks (CNN), recurrent neural networks (RNN) and multi-layer perceptrons (MLPs)). Support Vector Machines (SVM) and Auto-Regressive Itegrative Moving Average (ARIMA) appear frequently throughout the literature as well. Energy Demand Forecasting is time series forecasting problem. One effective and widely utilized time forecasting framework is FBProphet [8], introduced by Facebook in 2017. The advantages of FBProphet over ARIMA include faster fitting times, fitting time-series data with unevenly spaced intervals and the ability to accommodate seasonality with multiple periods.

2.3. Topic Modeling using LDA

Topic Modeling is a machine learning technique that enables algorithmic derivation of topics from unstructured text corpora [2]. The topic modeling technique of interest in our project is Latent Dirichlet Allocation (LDA) - a generative probabilistic model for unstructured collections of discrete data [1]. LDA is particularly useful in our context due to its

applicability in document modeling and text classification. More specifically, given a set of unstructured documents, LDA then clusters the words in a document to form topics. These topics are then representative of the document in question. A challenge in using topic modeling to approximate events of interest from social media data is that of causality. How can we be sure that a topic representing a cluster of words scraped from twitter has a meaningful relationship with a real world event and not simply a correlation. The authors in [2] also faced this problem, and showed that using LDA to generate topics from social media text data results in a statistically valid approximation of a real world event. This finding is of particular interest in this project, as we seek to use social media to connect events in the real world with electricity demand.
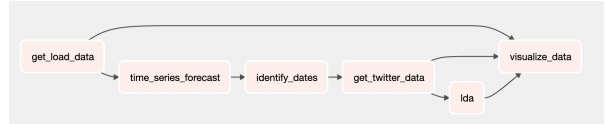
3. Progress

Figure 1. Overall System Architecture

Figure 1 shows the DAG of overall system architecture, which we will discuss each component further in each subsection.

3.1. Getting Electricity Load Data and Forecasting using FBProphet

To make the prediction, first we need to collect the data. We use Python `requests` library for API web scraping, collecting load data from NYISO Public Reports, then store the data in Google Cloud Storage as CSV files, and create a BigQuery database from the stored CSV files in Google Cloud Storage.
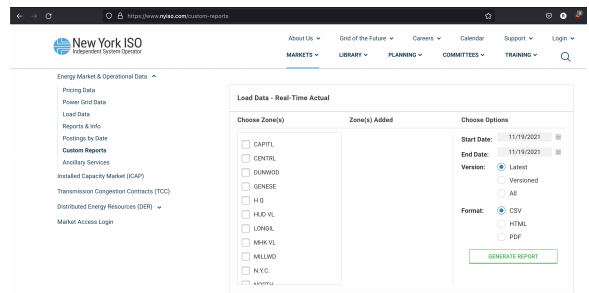

Figure 2. NYISO Data Source

After getting the electricity load dataset, we can then start to do the time-series forecasting model. We use `FBProphet` library for the forecasting. FBProphet [8] is a procedure for forecasting time series data based on an
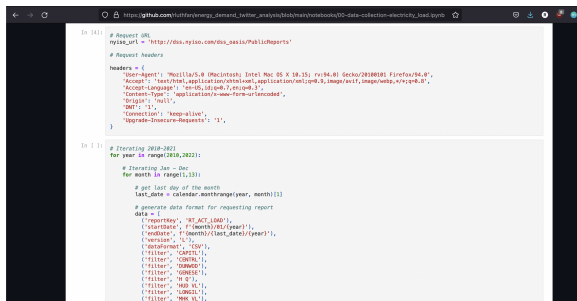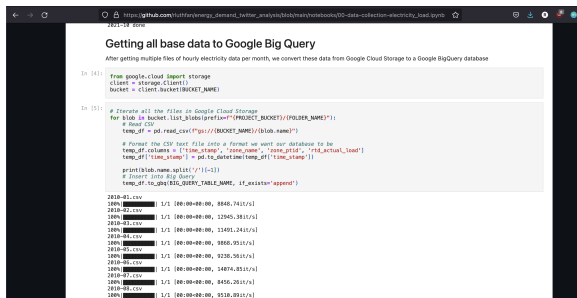
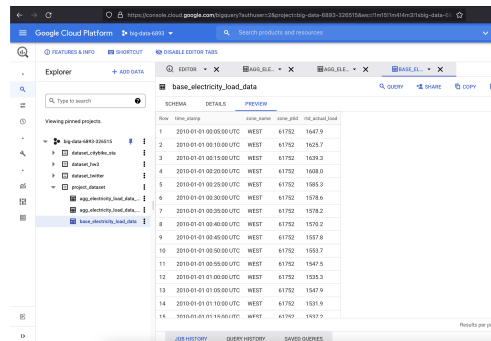Figure 3. Save to GCS

Figure 4. Save to BigQuery

Figure 5. BigQuery Database

additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

We use `pandas-gbq` library for querying load data from BigQuery. After getting Pandas DataFrame of electricity load, we convert it into a Spark DataFrame. Our goal is to generate not only one forecast for the entire dataset, but multiple models and forecasts for each zone, something that would be incredibly time-consuming to perform as a sequential operation. By leveraging a `Spark` cluster, individual worker nodes in the cluster can train a subset of models in parallel with other worker nodes, greatly reducing the overall time required to train the entire collection of time series models.

```python
# Snippet for querying data from BigQuery
sql_query = f"""
    SELECT
        zone_name
        , hourly_timestamp AS ds
        , SUM(sum_rtd_actual_load) AS y
    FROM {BIG_QUERY_TABLE_NAME}
    WHERE
        hourly_timestamp >= '2018-01-01'
        AND hourly_timestamp <= '2020-12-31'
        --AND zone_name = 'N.Y.C.'
    GROUP BY
        1,2
"""
load_df = pd.read_gbq(sql_query)

# Converting Pandas DataFrame to Spark RDD
spark_df = sqlContext.createDataFrame(load_df)
```

We will then use `pandas function` capability of Spark to apply a custom function to each group of data in our DataFrame. Within the function definition, we instantiate our model, configure it and fit it to the data it has received. The model makes a prediction, and that data is returned as the output of the function. We also use the same method for evaluating the prediction.

```python
### Modelling function

def forecast_load(history_pd: pd.DataFrame)
-> pd.DataFrame:
    # remove missing values
    # (more likely at high granularity level)
    history_pd = history_pd.dropna()

    # instantiate the model,
    # configure the parameters
    model = Prophet(
        changepoint_prior_scale=0.5,
        seasonality_mode='multiplicative',
        interval_width=0.95,
    )
    model.add_country_holidays(country_name='US')

    # fit the model
    model.fit(history_pd)

    # configure predictions
    future_pd = model.make_future_dataframe(
        periods=90,
        freq='d',
        include_history=True
    )

    # make predictions
    forecast_pd = model.predict(future_pd)

    # ASSEMBLE EXPECTED RESULT SET
    # ------------------------------------
    # get relevant fields from forecast
    col_f = [
        'ds','yhat', 'yhat_upper', 'yhat_lower'
    ]
    f_pd = forecast_pd[col_f]
    # get relevant fields from history
```

3

```python
    h_pd = history_pd[['ds','zone_name','y']]

    # left join
    results_pd = pd.merge(
        f_pd, h_pd, how='left', on=['ds']
    )

    # return predictions
    return results_pd
```

```python
# Use the modelling function
# as Spark aggregation function
result_schema =StructType([
    StructField('ds',DateType()),
    StructField('zone_name',StringType()),
    StructField('y',FloatType()),
    StructField('yhat',FloatType()),
    StructField('yhat_upper',FloatType()),
    StructField('yhat_lower',FloatType())
])

forecast = (
    spark_df
    .groupBy("zone_name")
    .applyInPandas(
        forecast_load, schema=result_schema
    )
)
forecast.createOrReplaceTempView('new_forecasts')
```


Figure 6. Forecast Results in Table Format and its Performance

The example of the prediction results can be seen in Figure 6 for tabular format, and Figure 8 for graph format. Using FBProphet, we can also see the seasonality and which component contributes more to the prediction, as it can be seen from Figure 7. This components are useful for interpreting our model and understanding how much each the
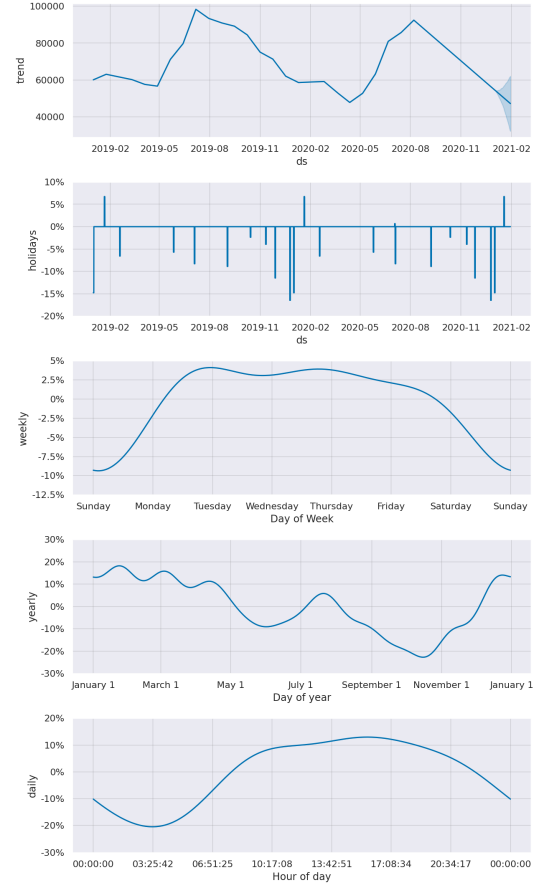

Figure 7. Forecast Components

trends, seasonalities (daily, weekly, monthly, yearly), and holidays contributes to overall energy demand. It is also useful for tuning our model parameters.

The next step for this part of the project is to do the hyperparameter tuning. We can see from the results in Figure 8 that there are some trend change points (indicated by vertical red line) still not yet captured by the model, as we can see some movement in the black dots (actual data) but the red trend line missed those. FBProphet model gives plenty of parameters to be tuned, such as `mcmc_samples` option to get uncertainty in seasonality (via Bayesian sampling) and also manually specifying the `period` and `fourier_order` of each seasonality, so we still need to figure out which metric will give us significant increase in performance.

When the performance metrics already satisfies the requirements, we will then use the forecast results to identify anomalous spikes and troughs, which we define as those dates which have actual load values with difference to the prediction bigger than our margin of error. These dates is then used for streaming Twitter data.
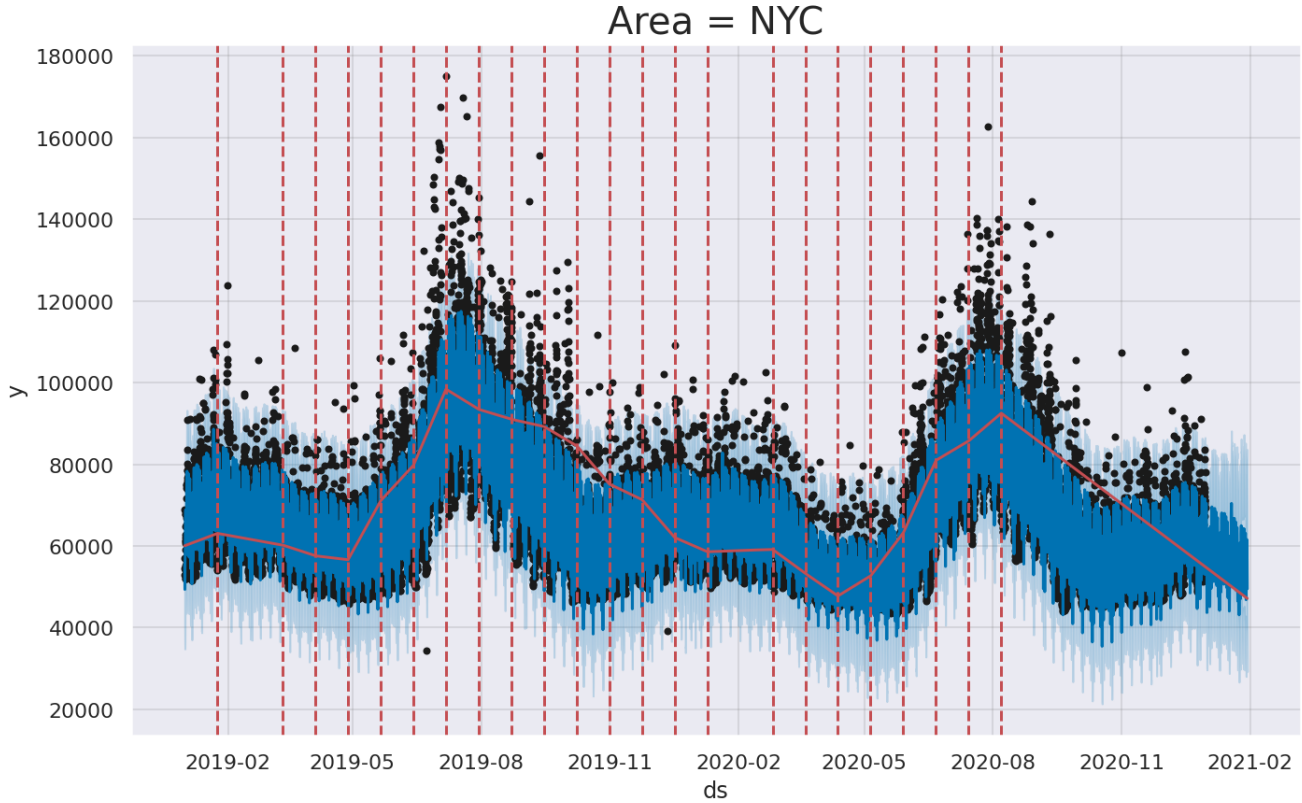
Figure 8. Energy Demand Forecast Results

3.2. Getting Twitter Data on Specific Dates & Locations

In order to collect historical data from Twitter, we had to apply and get approved for Twitter's Academic Research API which provides access to Twitter's Full-Archive Search, with access to tweets from 2006 to present, a higher monthly tweet cap, and enhanced features designed to support research [9]. The approval process consisted of submitting an application that described the project and our intentions with it. Once approved, we started streaming the Twitter data on significant dates that could possibly impact energy demand, such as on Christmas, fourth of July, New Year's day, etc., during the past 10 years. This data was then loaded onto Google Drive using GCP's Google Drive API to be accessible when performing the topic modelling.

As can be seen in Figure 9, the historical tweets are streamed using Twitter's Full-Archive search. A maximum of 500 tweets can be streamed in one GET request. The tweets can be filtered according to location, post date, user details, such as if the user is verified or not, etc. This powerful tool allows us to retrieve tweets and their accompanying fields, such as post location, post time, author ID, and more, into a json file to be stored on Drive as done in Figure 10.


```python
search_url = "https://api.twitter.com/2/tweets/search/all"

# Possible queries:
# 'place:"new york" OR 'place:"albany" OR place:"niagara" is:verified'

# Use the below links for reference on how to build queries:
# https://developer.twitter.com/en/docs/twitter-api/tweets/search/integrate/build-a-query#availability
# https://developer.twitter.com/en/docs/twitter-api/tweets/search/api-reference/get-tweets-search-all
# https://developer.twitter.com/en/docs/twitter-api/tweets/search/quick-start/full-archive-search

# Optional params: start_time,end_time,since_id,until_id,max_results,next_token,
# expansions,tweet.fields,media.fields,poll.fields,place.fields,user.fields
query_params = {'query': 'place:"manhattan, ny" OR place:"new york city" OR place:"brooklyn, ny" OR place:"bronx, ny" OR plac
                'tweet.fields': 'author_id,created_at',
                'max_results': '500',
                'start_time': '2021-02-01T00:00:01Z',
                'end_time': '2021-02-01T23:59:59Z',
                'place.fields': 'country_code',
                'expansions': 'geo.place_id'}


def bearer_oauth(r):
    """
    Method required by bearer token authentication.
    """

    r.headers["Authorization"] = f"Bearer {bearer_token}"
    r.headers["User-Agent"] = "v2FullArchiveSearchPython"
    return r


def connect_to_endpoint(url, params):
    response = requests.request("GET", search_url, auth=bearer_oauth, params=params)
    print(response.status_code)
    if response.status_code != 200:
        raise Exception(response.status_code, response.text)
    return response.json()

json_response = connect_to_endpoint(search_url, query_params)
print(json.dumps(json_response, indent=4, sort_keys=True))
with open('2021-02-01_nyc.json', 'w') as fp:
    json.dump(json_response, fp, indent=4, sort_keys=True)
```

Figure 9. Get Tweets using Twitter API Full-Archive Search

As one would expect, the earlier the date the lesser number of available tweets due to the fact that Twitter was only founded in 2006. Therefore, some records, mostly from 2010-2011, have little to no tweets even when the location was filtered for NYC, the location with the highest volume of tweets in NY State. An example of the data retrieved in NYC on the fourth of July, 2021 is shown in Figure 11.

5

Store Data on Google Drive

```python
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive

#gauth = GoogleAuth()
drive = GoogleDrive(gauth)

gauth = GoogleAuth()
# Try to load saved client credentials
gauth.LoadCredentialsFile("mycreds.txt")
if gauth.credentials is None:
    # Authenticate if they're not there
    gauth.LocalWebserverAuth()
elif gauth.access_token_expired:
    # Refresh them if expired
    gauth.Refresh()
else:
    # Initialize the saved creds
    gauth.Authorize()
# Save the current credentials to a file
gauth.SaveCredentialsFile("mycreds.txt")

upload_file_list = ['2021-02-01_nyc.json']
for upload_file in upload_file_list:
    gfile = drive.CreateFile({'parents': [{'id': '19t7W5_NMUl6A2XYPCbLIiiSr4XH2Zpsj'}]})
    # Read file and set it as the content of this instance.
    gfile.SetContentFile(upload_file)
    gfile.Upload() # Upload the file.
```

Figure 10. Store Data on Google Drive

Figure 11. Tweets Stored on Google Drive

```
{
    "data": [
        {
            "author_id": "232759962",
            "created_at": "2021-07-04t23:59:57.000z",
            "geo": {
                "place_id": "00c55f041e27dc51"
            },
            "id": "1411837215861362690",
            "text": "@gorilladrama @lakemonster2152 Catfish can taste with their whole body so just by fist bumping or shaking
        },
        {
            "author_id": "44133434",
            "created_at": "2021-07-04t23:59:57.000z",
            "geo": {
                "place_id": "011add077f4d2da3"
            },
            "id": "1411837213617364993",
            "text": "Lucky \ud83d\udc95\ud83d\udc95https://t.co/087DaqMxFP"
        },
        {
            "author_id": "2413170369",
            "created_at": "2021-07-04t23:59:55.000z",
            "geo": {
                "place_id": "01a9a39529b27f36"
            },
            "id": "1411837205891399680",
            "text": "@BonniethePuppy May i boop your nose?"
        },
        {
            "author_id": "152448004",
            "created_at": "2021-07-04t23:59:47.000z",
            "geo": {
                "place_id": "01a9a39529b27f36"
            },
            "id": "1411837173620514817",
            "text": "You pushing buttons with this one https://t.co/KdaZOIhxtx"
        },
```

Figure 12. Tweets from NYC on 07/04/2021

As can be seen in Figure 12, the information we are collecting includes the body of the tweet (to perform topic modelling on), the location the tweet was sent from (to identify the user location to correlate with energy demand in that location), and the author ID (to ensure that we are not collecting tweets from just a few users). The Twitter API allows us to collect more information about each tweet, however, as of now, the fields identified prove to be sufficient for our planned experiments. Once all dates with anomalous

demand spikes and troughs are identified by FBProphet, the tweets on those dates can be streamed using the aforementioned system.

4. Planned Experiments

4.1. Topic Modelling on Twitter Data

We will use large-scale social media network data as method of tracking a subset of events that are relevant to the social media network users, in this particular case is electricity energy usage. Let the events be X, the hypothesis is that if a social media network user is exposed to a particular event $x_i \in X$, this event may induce a user to post a message m at the time t the event is happening. The message m_t is assumed to be text based, in which it can be represented with a word vector v_t.

It is actually easy for the social media network user to map $x_i \rightarrow m_t$, for example a user might want to tweet about a football player after watching a football match. However, to reverse this mapping in a machine processable manner is hard, hence we must approach this mapping in an indirect fashion. We will develop topic models using LDA from the word vectors v_t of the tweets streamed, where we assume a topic o is an approximation to event x_i for some i.

To determine the event (tweet topics) and phenomenon (electricity load spikes/troughs) causality, we will first use Pearson's rank correlation where each point is a pairing of event frequencies and real world phenomena to compare the event's time series to the document time series related to the real-world phenomena (such as holidays). We should not filter by positive or negative correlation, since a strong negative relationship between an event and a real world event can be just as interesting as a positive one. The next step is then to figure out which correlation implies causal link. We will use domain expertise to verify tweet to energy spikes causation.

4.2. Data Visualization via a Web Dashboard

Once all data is collected, topic modelling done, impact topics identified, domain expertise used to determine if the topics identified are merely correlations or meaningful causal relationships, we will build a web dashboard to display our results. Figure 13 below shows a preliminary layout of the dashboard without the Twitter/LDA data incorporated.

The dashboard is in a user query-based format where the electricity load data will be granular up to an hour with an adjustable time scale for the user to decrease granularity down to a yearly display. The user can also filter the data by regions in NY state. The topics identified using LDA from the Twitter data will be added on to the dashboard and given a scale value for impact, i.e. High, Medium, and Low impact. This will be determined not only from the topic
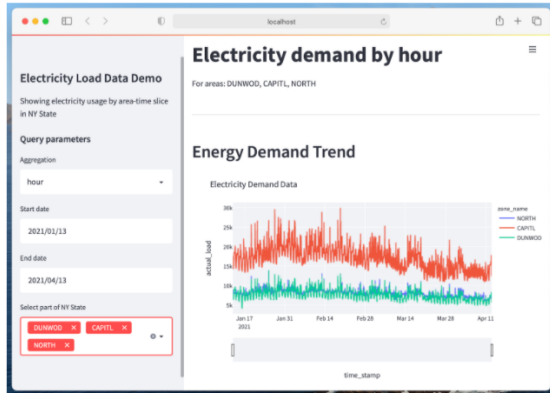
Figure 13. Preliminary Data Visualization on a Web Dashboard (without Twitter Data)

modelling but from power systems domain expertise. The final dashboard will be shown in the bottom figures with various user queries and results.

Figure 14. Placeholder Data Visualization on a Web Dashboard (with Twitter Data query 1)

Figure 15. Placeholder Data Visualization on a Web Dashboard (with Twitter Data query 2)

As will be seen in the Figures 14 and 15 above, for an example date (yet to be identified), we will display the electricity load data of that region in NY state with accompanying information regarding the corresponding social media activity. If we deem a Twitter topic to be High impact we will mark it accordingly, possibly via color coordination, to show a strong meaningful causal relationship with the energy spike. Medium impact events will be marked to show a possible causal relationship with energy demand. Finally, Low impact events will be marked to indicate a improbable causal relationship/mere correlation.

We hope our dashboard will tie our findings into an easily digestible format for the average person without prior knowledge of topic modelling, and electricity market and social media analysis, to be able to understand how Twitter use could be an indicator of real-world events and possibly drive energy demand. Additionally, we also aim to provide value to electric utilities and independent system operators by giving them the strategic ability to manage energy use from impact topics determined from social media.

References

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003. 2

[2] Todd Bodnar, Matthew L. Dering, Conrad Tucker, and Kenneth M. Hopkinson. Using large-scale social media networks as a scalable sensing system for modeling real-time energy utilization patterns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(10):2627–2640, 2017. 1, 2

[3] Olivier Kraaijeveld and Johannes De Smedt. The predictive power of public twitter sentiment for forecasting cryptocurrency prices. *Journal of International Financial Markets, Institutions and Money*, 65:101188, 2020. 1

[4] Ana Luna, Miguel Nuñez-del Prado, Alvaro Talavera, and Erick Somocurcio Holguín. Power demand forecasting through social network activity and artificial neural networks. In *2016 IEEE ANDESCON*, pages 1–4, 2016. 1, 2

[5] Isaac Kofi Nti, Moses Teimeh, Owusu Nyarko-Boateng, and Adebayo Felix Adekoya. Electricity load forecasting: a systematic review. *Journal of Electrical Systems and Information Technology*, 7(1):1–19, 2020. 2

[6] Mourad Oussalah and Ahmed Zaidi. Forecasting weekly crude oil using twitter sentiment of u.s. foreign policy and oil companies data. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 201–208, 2018. 1

[7] J. O. Petinrin and Mohamed Shaaban. Smart power grid: Technologies and applications. In *2012 IEEE International Conference on Power and Energy (PECon)*, pages 892–897, 2012. 2

[8] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018. 2

[9] Twitter. Twitter api - academic research access, 2021. 5