EECS E6893 Big Data Analytics Project 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 analytics. 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 [4] and crude oil price prediction [7]. 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 [5]. 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 [5]. 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 [6]. 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 [8]. 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 [6]. 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 [9], 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. Data

We use three types of data:

1. Time-series electricity load data from NYISO
   (a) Across 15 regions in NY state
   (b) 5 minutes granularity
   (c) From 01/01/2010 to 12/31/2020 for training, 01/01/2021 to 10/31/2021 for validation
2. Forecast results data from FBProphet
   (a) Produces energy demand prediction
   (b) Using historical comparison we will identify anomalous dates
3. Streamed Twitter tweets data
   (a) Region-level top/trending tweets with timestamps
   (b) Includes pictures/videos, likes and retweets

These datasets satisfy the 3V's of big data:

1. Volume: Data gathered from multiple sources, within a long time period
2. Velocity: Mixed velocity among the datasets: Twitter (fast flow), load (batches)
3. Variety: 3 distinct datasets, new and existing

4. System Overview

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

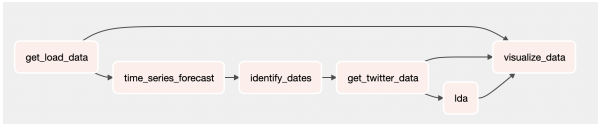


Figure 1. Overall System Architecture

5. Methods

5.1. 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.

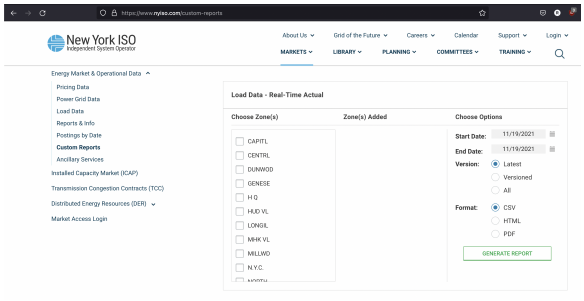


Figure 2. NYISO Data Source

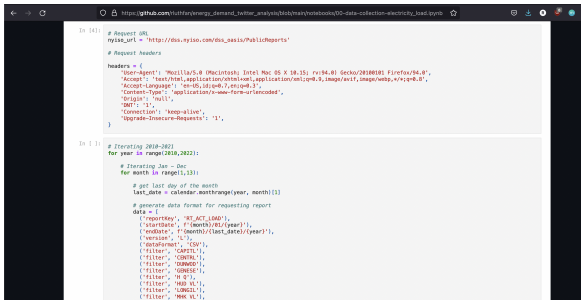


Figure 3. Save to GCS

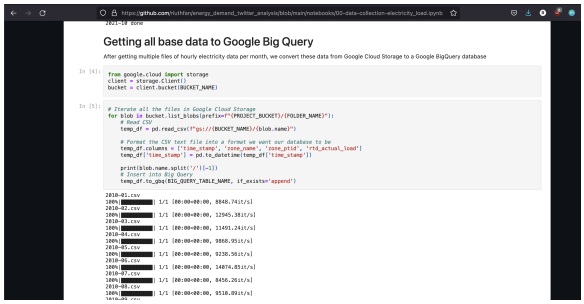


Figure 4. Save to BigQuery

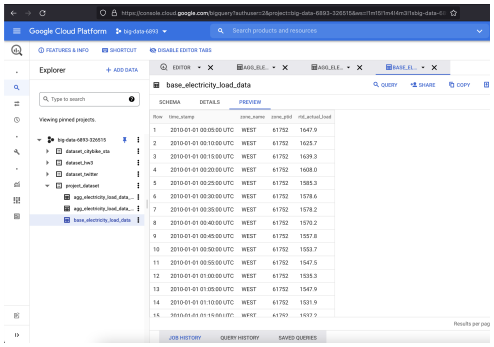


Figure 5. BigQuery Database

After getting the electricity load dataset, we then start to do the time-series forecasting model. We use `FBProphet` library for the forecasting. FBProphet [9] does the forecasting by either additive or multiplicative model, and adjust trends based on seasonalities of all types (regular daily, weekly, yearly and also irregular holidays). This way, the library is robust to trends shift and outliers.

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. Spark DataFrame is used to generate multiple models in parallel to reduce the time required for time-series model training.

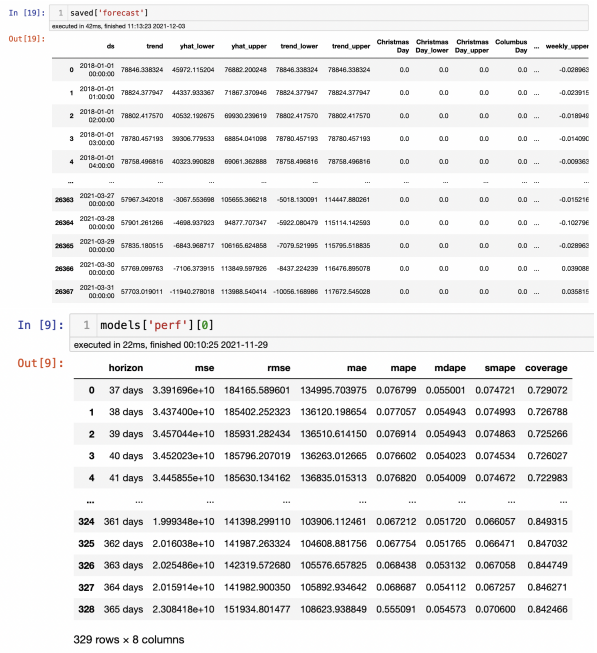
```
# Snippet for querying data from BigQuery
sql_query = f"""
    SELECT
        zone_name
        , hourly_timestamp AS ds
        , SUM(sum_rtd_actual_load) AS y
    FROM {BIG_QUERY_TABLE_NAME}
    WHERE
        hourly_timestamp >= '2018-01-01'
        AND hourly_timestamp <= '2020-12-31'
        --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 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.

```
### Modelling function

### Modelling function

def forecast_load(history_pd: pd.DataFrame)
-> pd.DataFrame:
```

```
    # remove missing values
    history_pd = history_pd.dropna()

    # instantiate the model & 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=24*30,
        freq='H',
        include_history=True
    )

    # make predictions
    forecast_pd = model.predict(future_pd)


    # ASSEMBLE EXPECTED RESULT SET
    # --------------------------------------
    # get relevant fields from forecast
    f_pd = forecast_pd[[
        'ds','yhat', 'yhat_upper',
        'yhat_lower', 'trend', 'holidays',
        'yearly', 'daily', 'weekly'
    ]]
    # get relevant fields from history
    h_pd = history_pd[['ds','zone_name','y']]

    # left join
    results_pd = pd.merge(f_pd, h_pd,
                          how='left', on=['ds'])
    results_pd['zone_name'] =
    (results_pd['zone_name']
    .fillna(method="ffill"))

    # return predictions
    return results_pd
```

```
# Schema of FBProphet Results
result_schema =StructType([
    StructField('ds', TimestampType()),
    StructField('zone_name', StringType()),
    StructField('y', FloatType()),
    StructField('yhat', FloatType()),
    StructField('yhat_upper', FloatType()),
    StructField('yhat_lower', FloatType()),
    StructField('trend', FloatType()),
    StructField('holidays', FloatType()),
    StructField('yearly', FloatType()),
    StructField('daily', FloatType()),
    StructField('weekly', FloatType()),
])

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



In [19]:
```
saved['forecast']
```
executed in 42ms, finished 11:13:23 2021-12-03

Out[19]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	Christmas Day	Christmas Day_lower	Christmas Day_upper	Columbus Day	...	weekly_upper
0	2018-01-01 00:00:00	78846.338324	45972.115204	76882.200248	78846.338324	78846.338324	0.0	0.0	0.0	0.0	...	-0.028963
1	2018-01-01 01:00:00	78824.377947	44337.933367	71867.370946	78824.377947	78824.377947	0.0	0.0	0.0	0.0	...	-0.023915
2	2018-01-01 02:00:00	78802.417570	40532.192675	69930.239619	78802.417570	78802.417570	0.0	0.0	0.0	0.0	...	-0.018949
3	2018-01-01 03:00:00	78780.457193	39306.779533	68854.041098	78780.457193	78780.457193	0.0	0.0	0.0	0.0	...	-0.014090
4	2018-01-01 04:00:00	78758.496816	40323.990828	69061.362888	78758.496816	78758.496816	0.0	0.0	0.0	0.0	...	-0.009363
...	...	...	...	...	...	...	...	...	...	...	...	...
26363	2021-09-27 00:00:00	57967.342018	-3067.553698	105655.366218	-5018.130091	114447.880261	0.0	0.0	0.0	0.0	...	-0.015216
26364	2021-09-28 00:00:00	57901.261266	-4698.937923	94877.707347	-5922.080479	115114.142593	0.0	0.0	0.0	0.0	...	-0.102796
26365	2021-09-29 00:00:00	57835.180515	-6843.968717	106165.624858	-7079.521995	115795.518835	0.0	0.0	0.0	0.0	...	-0.028963
26366	2021-09-30 00:00:00	57769.099763	-7106.373915	113849.597926	-8437.224239	116476.895078	0.0	0.0	0.0	0.0	...	0.039088
26367	2021-09-31 00:00:00	57703.019011	-11940.278018	113988.540414	-10056.168986	117672.545028	0.0	0.0	0.0	0.0	...	0.035815

In [9]:
```
models['perf'][0]
```
executed in 22ms, finished 00:10:25 2021-11-29

Out[9]:

	horizon	mse	rmse	mae	mape	mdape	smape	coverage
0	37 days	3.391696e+10	184165.589601	134995.703975	0.076799	0.055001	0.074721	0.729072
1	38 days	3.437400e+10	185402.252323	136120.198654	0.077057	0.054943	0.074993	0.726788
2	39 days	3.457044e+10	185931.282434	136510.614150	0.076914	0.054943	0.074863	0.725266
3	40 days	3.452023e+10	185796.207019	136263.012665	0.076602	0.054023	0.074534	0.726027
4	41 days	3.445855e+10	185630.134162	136835.015313	0.076820	0.054009	0.074672	0.722983
...	...	...	...	...	...	...	...	...
324	361 days	1.999348e+10	141398.299110	103906.112461	0.067212	0.051720	0.066057	0.849315
325	362 days	2.016038e+10	141987.263324	104608.881756	0.067754	0.051765	0.066471	0.847032
326	363 days	2.025486e+10	142319.572680	105576.657825	0.068438	0.053132	0.067058	0.844749
327	364 days	2.015914e+10	141982.900350	105892.934642	0.068687	0.054112	0.067257	0.846271
328	365 days	2.308418e+10	151934.801477	108623.938849	0.555091	0.054573	0.070600	0.842466

329 rows × 8 columns

Figure 6. Forecast Results in Table Format and its Performance

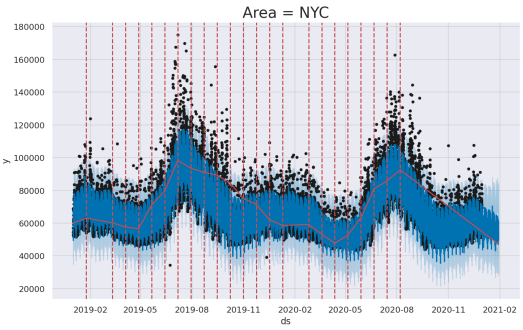


Figure 7. Energy Demand Forecast Results

The example of the prediction results can be seen in Figure 6 for tabular format, and Figure 7 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 8. This components are useful for interpreting our model and understanding how much each the 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 7 that there are some trend change points (indicated by vertical red line) still not yet captured by the

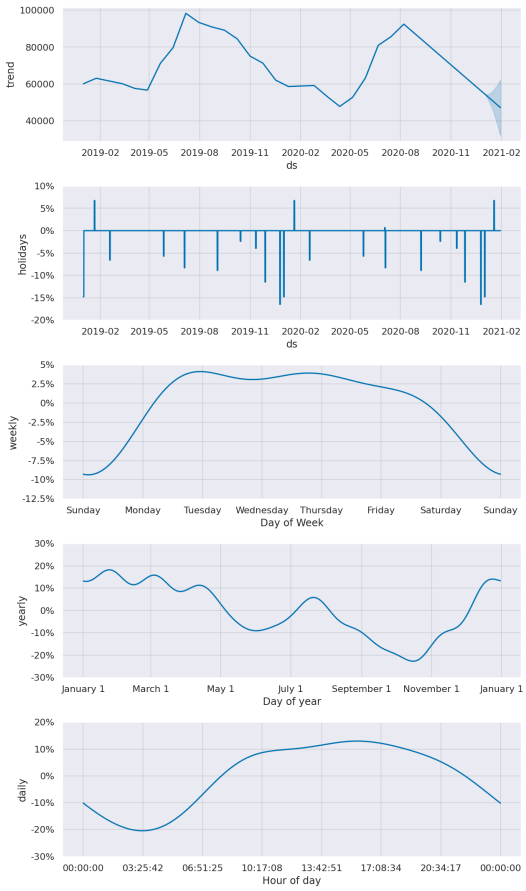


Figure 8. Forecast Components

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.

5.2. 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 [10]. 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.

```
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 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.

Store Data on Google Drive

```
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

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

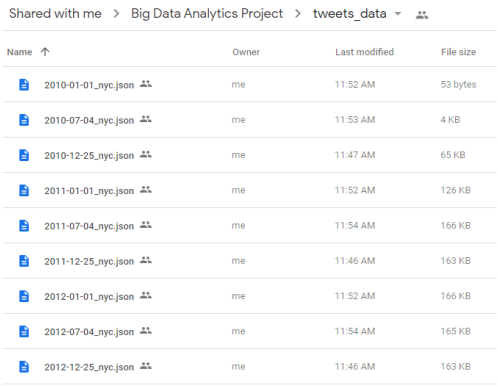


Figure 11. Tweets Stored on Google Drive

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.

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.

5.3. Topic Modelling with LDA

After getting tweets data, we then do the LDA topic modelling. The tweets is stored in a storage, and we store each date as a different JSON file. We did not create a BigQuery database for tweets data because it is unstructured. We use Python's `json` library to load tweets iteratively. This loaded JSON file is then appended into a `pandas DataFrame` to be used in the next step, which is text preprocessing. For preprocessing, we used Python `NLTK` and `gensim` library. We remove Links, remove Usernames, remove Hashtags, text Lemmatization and Tokenization vectorization. Then, we do topic modelling with LDA using `gensim` library with hyperparameter number of topics.

6. Experiments

6.1. Topic Modelling on Twitter Data

We will use the collected Twitter tweets for checking their effects on real-world electricity energy usage. The hypothesis is that social media trends, in this case Twitter trending tweets, are induced by real-world event that might cause a change in electricity load demand. Based on experiments done by [2], a social media user that is exposed to an event x might be induced to post a message m at the time the event is happening. Their assumption is that the message is text-based, our Twitter tweets fits that assumption and it might be true that Twitter users may be induced to post a tweet because of an event that caused electricity demand spike and trough.

The first challenge is doing a reverse mapping, as is also stated by [2]. As the social media user, the person knows that an event is happening and wants to tweet about it. But what we have is tweets data after-the-fact. To tackle this issue, we must use indirect approach. The approach is by using LDA [1] to group the tweets streamed into topics. These topics are used as an approximation of some event happening. Our assumption is that when the event that cause a spike or drop in electricity demand is big enough, more Twitter users will tweet about it, which means that the topic they are tweeting should be similar. Because more users are tweeting about a similar topic, the LDA model will be more robust.

The next challenge is to determine which tweet topics are causing electricity load spikes/troughs. We will use domain expertise to verify causality relationship between the topics generated from LDA and the spikes/troughs in electricity load time series.

6.2. 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 built a web dashboard to display our results. Figure 13 and 14 below shows the layout of the dashboard.

The dashboard is in a user query-based format where the electricity load data is 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 is added on to the dashboard and given a scale

Figure 13. Electricity Load Data Visualization on a Web Dashboard

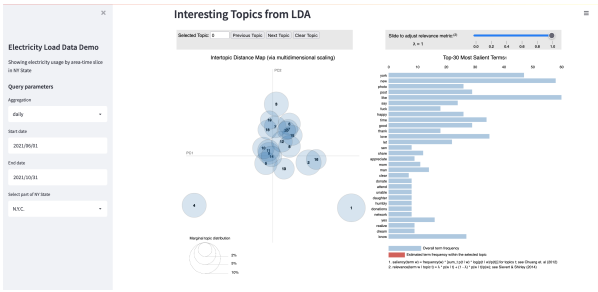


Figure 14. LDA Topics Data Visualization on a Web Dashboard

value for impact, which is visualized using bubble chart and bar chart. This is determined not only from the topic modelling but from power systems domain expertise.

This dashboard is built using Python `streamlit` library, with the line chart of electricity load data is plotted using `plotly` library and LDA topic modelling results is plotted using `pyLDAvis` library. 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.

7. Key Results and Conclusion

After performing the LDA analysis on the anomalous dates, the results we got included many irrelevant topics. Therefore, using domain expertise, we had to manually filter out the topics that could realistically cause electricity usage spikes. These three categories of topics are also not currently being accounted for in demand prediction models by utilities, therefore, this is a novel discovery.

Firstly, the topic we saw the most was related to traffic and accidents. They showed up mostly in car-heavy regions of NY state like Albany and Buffalo. Surprisingly, NYC had nothing to do with traffic probably due to the prevalence of public transportation. This topic is relevant because more traffic means more accidents, which strains emergency resources like police and hospitals. This naturally increases electricity use. Also, traffic congestion causes people to work later which means buildings have to be open longer and use more energy. Figure 15 shows the LDA results for the CAPITAL load zone, which includes the NY state capital of Albany. As can be seen, there are many irrelevant results, such as 'neurology', 'thanks', etc. However, using domain expertise we were able to determine that the most relevant topics were traffic/accident related.

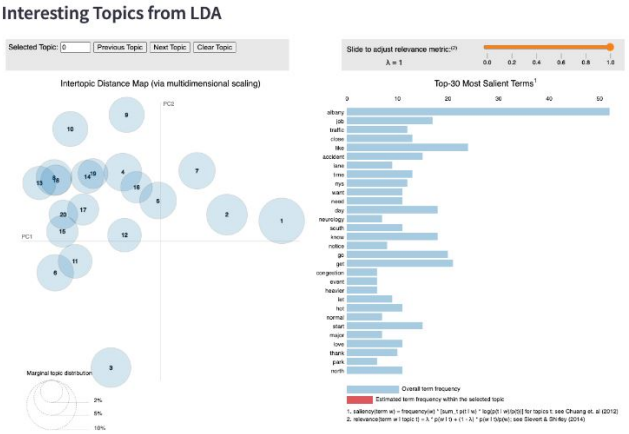


Figure 15. LDA Results of CAPITAL Zone (Albany)

Secondly, another frequent and relevant topic we encountered was to do with construction, both regular and road-related. Construction is naturally electricity-intensive, therefore, this was a straightforward conclusion to make. Additionally, road construction is not only electricity intensive in and of itself, but has the potential to cause road accidents as evidenced in reference [3]. Therefore, due to the reasons mentioned above, concluding that construction could result in energy spikes was reasonable. Figure 16 shows, once again, that we had to filter out many irrelevant topics to determine that construction was a relevant topic that affected energy usage.

Lastly, a category of topics that prominently showed up in New York City, especially in the last 5-6 years was cultural events. Cultural events like sports games, such as Yankees baseball games, could realistically affect electricity use. This is because people have watch parties for games, which at volume can cause an electricity spike. Additionally, people tend to flock to sports bars and restaurants which can affect electricity usage. Moreover, stadiums themselves are massive electrical loads, especially when a game is occurring. Another cultural event which affected NYC was New York Fashion Week. This is a week-long

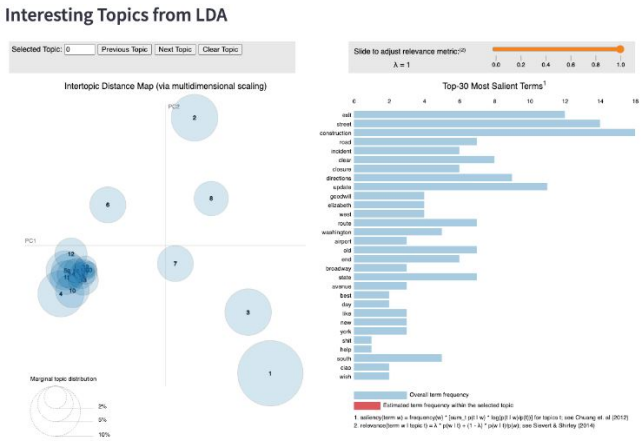


Figure 16. LDA Results of WEST Zone (Buffalo)

event that attracts celebrities and fashion designers, in addition to tourists. Therefore, NYFW is a natural choice when determining what types of topics affect electricity use in NYC. One thing to note is that, in NYC over the last 5-6 years, there was a proliferation of topics to do with tourism, such as 'photos', 'brooklyn bridge', etc. Therefore, we had to filter these out as they were irrelevant to our needs.

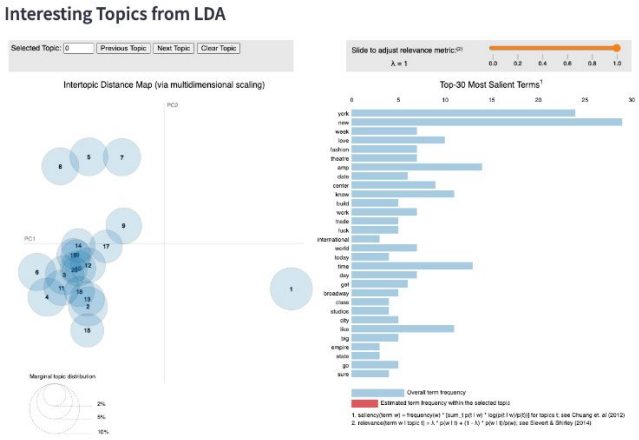


Figure 17. LDA Results of NYC Zone

Figure 17 shows the LDA results for NYC and as mentioned before there are many irrelevant tourism-related topics which are not relevant when discussing anomalous energy spikes. This is because NYC always has tourists, therefore, the presence of tourism-related topics is not probable to be a factor in *anomalous* energy spikes.

Overall, our topic modeling was done on the anomalous dates gathered from the FBProphet time-series forecasting and therefore is limited to those time frames. Expanding the number of dates and gathering Twitter data from more days could paint a bigger picture on topics that affect electricity usage. Additionally, the filtering of relevant topics was done manually, using domain expertise. However, in future iterations this process could be automated by utilizing a machine learning algorithm that is able to distinguish relevant topics that are correlated to spikes in energy demand, from other topics.

The reason this work is important is because the efficient prediction of anomalous spikes in energy demand is invaluable to electric utility companies. It allows adequate resource allocation and pricing. Additionally, it allows for fewer losses during unexpected demand spikes due to topics not accounted for in demand prediction models, such as traffic incidents, construction, and large cultural events.

We have shown that through the use of big data analytics, a social media sensing pipeline that monitors Twitter activity for traffic incidents or large cultural events could be utilized by utility companies to reduce losses incurred by anomalous spikes in demand.

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, 6

[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, 6

[3] Gourav Goel and S.N. Sachdeva. Analysis of road accidents on nh-1 between rd 98km to 148km. *Perspectives in Science*, 8:392–394, 2016. Recent Trends in Engineering and Material Sciences. 7

[4] 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

[5] 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

[6] 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

[7] 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

[8] 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

[9] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018. 2, 3

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