# Search for a Connection: Energy Demand & Twitter Trending Topics

**EECS 6893**
**Big Data Analytics**
**Final Project Report**

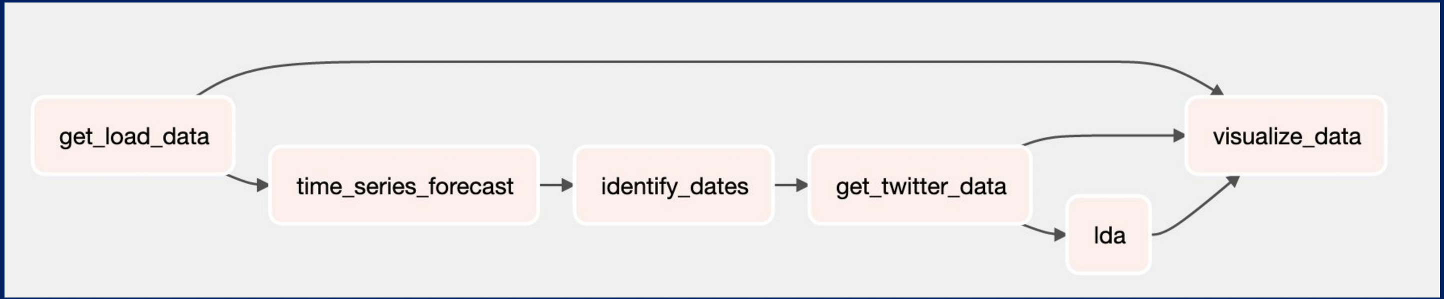
**Group:**
**Kevin Mark Murning (kmm2344)**
**Rifqi Luthfan (rl3154)**
**Rohan Raghuraman (rr3417)**

# Contents

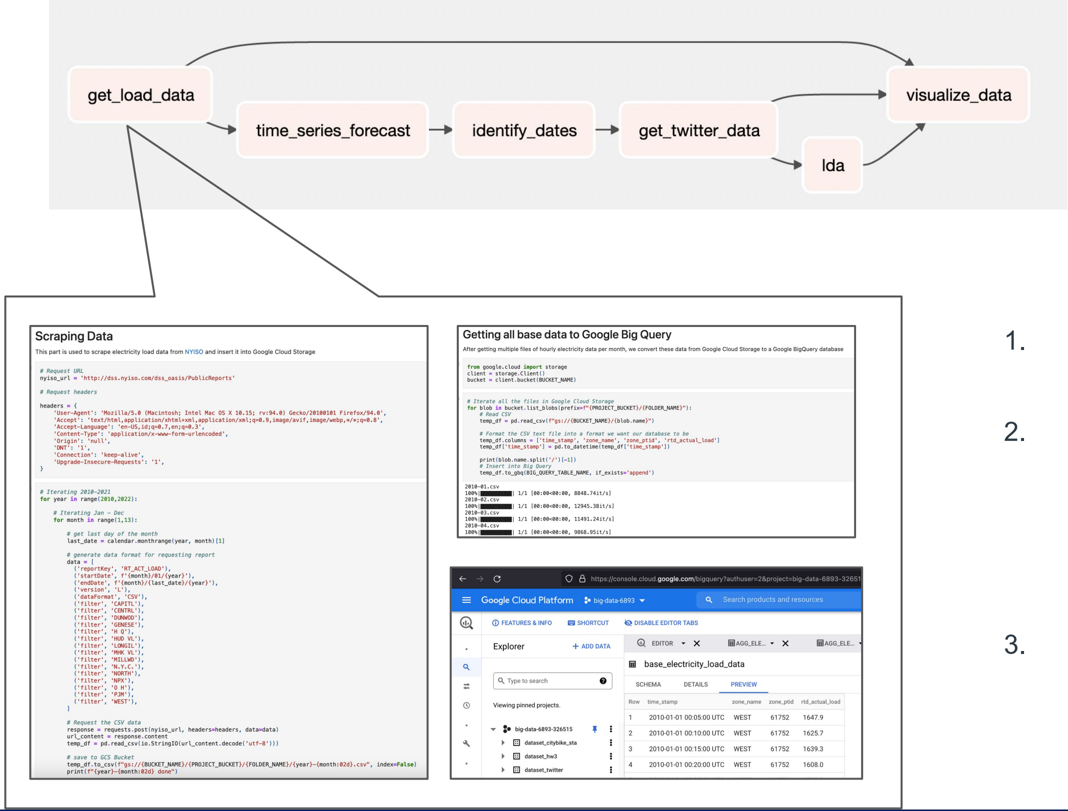
# Motivation

- **Effective energy demand forecasting plays a vital role in power systems**
  - Resource Allocation
  - Economically viable pricing
- **Some trends are easy to infer**
  - 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**
  - How can we predict anomalous changes in demand?
- **Much work has been done on monitoring social media for time series forecasting**
  - Very little has been done in the energy sector
- **Our Question:**
  - *Can Twitter activity formulated into topics be linked in a causal relationship with energy demand spikes?*

# System, Experiment Design and Methodology

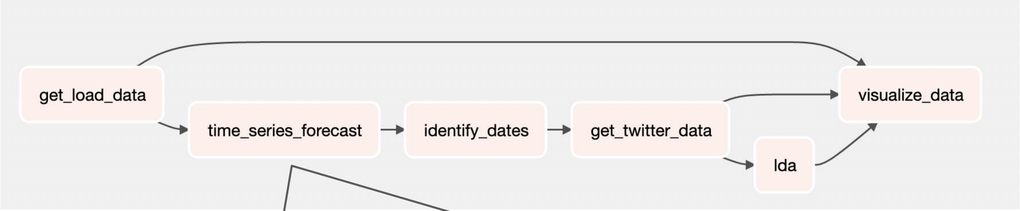


# Get Electricity Load Energy Demand Data




1. Web scraping: load data 5 mins granularity from [http://dss.nyiso.com/dss_oasis/PublicReports](http://dss.nyiso.com/dss_oasis/PublicReports) using python requests -> store in Google Cloud Storage as CSV files
2. Create BigQuery database from CSV files in Google Cloud Storage:
   a. base table -> store data as is
   b. Preprocessing data for aggregation table (hourly, daily, weekly, monthly, yearly) -> store data with the needed aggregation so we do not need to always run a heavy query -> used for web data viz
3. **Metrics that we track: size of database**
   a. Hourly aggregation table only take 25 MB of query compared to multiple GBs of raw data
   b. Daily weekly monthly yearly aggregation take at max 1 MB query size

# Time Series Forecast




1. Query from BigQuery using pandas-gbq and convert queried data into Spark RDD
2. Use FBProphet Library for forecasting
   a. Applied forecasting in parallel for different zones in NY State utilizing PySpark's applyInPandas()
3. **Metrics that we track: MAPE, RMSE**
   a. Target MAPE below 10%
   b. RMSE value is used for the next step
4. **Hyperparameters tuned:** changepoint_prior_scale, seasonality_mode, interval_width

Google 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 in ('N.Y.C.', 'CAPITL')
    */
    GROUP BY
        1,2
"""
load_df = pd.read_gbq(sql_query)
```

```
load_df['ds'] = pd.to_datetime(load_df['ds']).apply(lambda x: x.replace(tzinfo=None))

load_df
```

PySpark

```
## Convert to Spark RDD
spark_df = sqlContext.createDataFrame(load_df)
```

PROPHET

## Modelling

```
### 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=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
```

## Training loop

### Get Forecast Results

```
# 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')
```

```
forecast_df = forecast.toPandas()
```

```
21/12/09 17:31:47 WARN org.apache.spark.scheduler.TaskSetManager: Stage 0 contains a task of very large size (12282 KiB). The maxim
mmended task size is 1000 KiB.
```

```
forecast_df
```

| | ds | zone_name | y | yhat | yhat_upper | yhat_lower | trend | holidays | yearly | daily | weekly |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2010-01-01 00:00:00 | CAPITL | 12518.099609 | 12613.607422 | 16561.296875 | 9120.713867 | 14812.665039 | -0.095238 | 0.056130 | -0.130641 | 0.021291 |
| 1 | 2010-01-01 01:00:00 | CAPITL | 13092.299805 | 11989.854492 | 15666.069336 | 8186.646973 | 14813.125000 | -0.095238 | 0.056180 | -0.172363 | 0.020830 |
| 2 | 2010-01-01 02:00:00 | CAPITL | 12562.200195 | 11588.465820 | 15455.477539 | 7901.891602 | 14813.585938 | -0.095238 | 0.056229 | -0.199018 | 0.020314 |
| 3 | 2010-01-01 03:00:00 | CAPITL | 12101.299805 | 11466.584961 | 15152.679688 | 7877.807129 | 14814.046875 | -0.095238 | 0.056278 | -0.206743 | 0.019738 |
| 4 | 2010-01-01 04:00:00 | CAPITL | 11989.799805 | 11715.201172 | 15239.740234 | 8260.172852 | 14814.506836 | -0.095238 | 0.056327 | -0.189391 | 0.019095 |

## Forecast Evaluation

```
eval_df = forecast_df.groupby('zone_name').apply(evaluate_forecast).reset_index(drop=True)
```

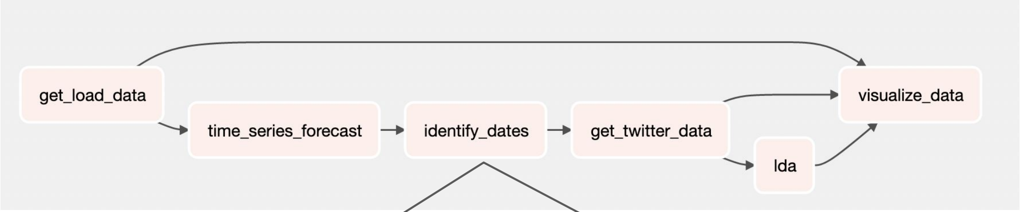
```
eval_df
```

| | training_date | zone_name | mae | mse | mape | rmse |
|---|---|---|---|---|---|---|
| 0 | 2021-12-09 | CAPITL | 1338.881470 | 3.543972e+06 | 0.079469 | 1882.543945 |
| 1 | 2021-12-09 | CENTRL | 1629.996704 | 5.235488e+06 | 0.072010 | 2288.118896 |
| 2 | 2021-12-09 | DUNWOD | 801.376038 | 1.314109e+06 | 0.092441 | 1146.345947 |
| 3 | 2021-12-09 | GENESE | 1127.091919 | 2.572215e+06 | 0.080100 | 1603.812744 |
| 4 | 2021-12-09 | HUD VL | 1307.080322 | 3.498391e+06 | 0.093496 | 1870.398682 |
| 5 | 2021-12-09 | LONGIL | 3057.967041 | 1.947232e+07 | 0.100375 | 4412.745605 |
| 6 | 2021-12-09 | MHK VL | 874.259399 | 1.425834e+06 | 0.079860 | 1194.082764 |
| 7 | 2021-12-09 | MILLWD | 464.994049 | 4.078183e+05 | 0.117714 | 638.606567 |
| 8 | 2021-12-09 | N.Y.C. | 5754.780762 | 6.861935e+07 | 0.077383 | 8283.679688 |
| 9 | 2021-12-09 | NORTH | 356.260620 | 2.901103e+05 | 0.049971 | 538.618896 |
| 10 | 2021-12-09 | WEST | 1426.481567 | 4.100283e+06 | 0.065248 | 2024.915527 |



Image source for logos: Google

# Identify Critical Dates




1. Use the forecast results and evaluation data
    a. Merge forecast and evaluation result, calculate forecast error of each date
    b. Get absolute difference value
2. Analyze trends & seasonality
3. **Create a metric to identify anomalous date based on analysis**
    a. RMSE value is used -> anomalous hour if the difference in actual vs predicted value is more than 5xRMSE value
    b. Anomalous date if 4 or more anomalous hours occurs in a day

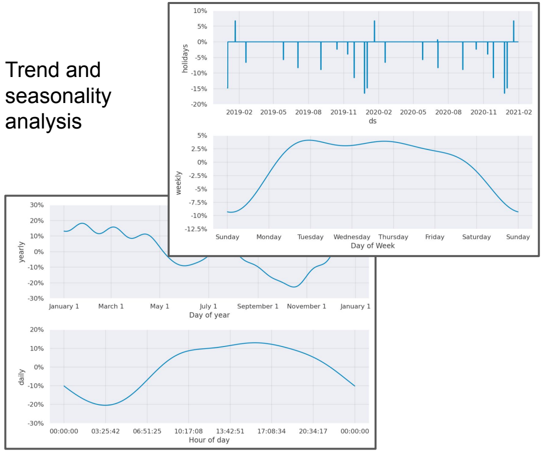
Merge forecast and evaluation result, calculate forecast error of each date

```
dates_df = pd.merge(
    forecast_df[(forecast_df['ds']<'2021-11-01')][['ds', 'zone_name', 'y', 'yhat']],
    eval_df[['zone_name', 'mae', 'rmse', 'mape']],
    on=['zone_name'], how='left'
)
dates_df['y_abs_diff'] = np.abs(dates_df['y']-dates_df['yhat'])
dates_df['y_pct_diff'] = dates_df['y_abs_diff']/dates_df['y']

dates_df
```

| | ds | zone_name | y | yhat | mae | rmse | mape | y_abs_diff | y_pct_diff |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2010-01-01 00:00:00 | CAPITL | 12518.099609 | 12613.607422 | 1338.881470 | 1882.543945 | 0.079469 | 95.507812 | 0.007630 |
| **1** | 2010-01-01 01:00:00 | CAPITL | 13092.299805 | 11989.854492 | 1338.881470 | 1882.543945 | 0.079469 | 1102.445312 | 0.084206 |
| **2** | 2010-01-01 02:00:00 | CAPITL | 12562.200195 | 11588.465820 | 1338.881470 | 1882.543945 | 0.079469 | 973.734375 | 0.077513 |
| **3** | 2010-01-01 03:00:00 | CAPITL | 12101.299805 | 11466.584961 | 1338.881470 | 1882.543945 | 0.079469 | 634.714844 | 0.052450 |
| **4** | 2010-01-01 04:00:00 | CAPITL | 11989.799805 | 11715.201172 | 1338.881470 | 1882.543945 | 0.079469 | 274.598633 | 0.022903 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1140871** | 2021-10-31 19:00:00 | MILLWD | 3901.063965 | 4167.789062 | 464.994049 | 638.606567 | 0.117714 | 266.725098 | 0.068372 |
| **1140872** | 2021-10-31 20:00:00 | MILLWD | 3805.951416 | 4060.871826 | 464.994049 | 638.606567 | 0.117714 | 254.920410 | 0.066979 |
| **1140873** | 2021-10-31 21:00:00 | MILLWD | 3647.767578 | 3807.325439 | 464.994049 | 638.606567 | 0.117714 | 159.557861 | 0.043741 |
| **1140874** | 2021-10-31 22:00:00 | MILLWD | 3449.541748 | 3461.595947 | 464.994049 | 638.606567 | 0.117714 | 12.054199 | 0.003494 |
| **1140875** | 2021-10-31 23:00:00 | MILLWD | 3095.496094 | 3104.627197 | 464.994049 | 638.606567 | 0.117714 | 9.131104 | 0.002950 |

Trend and seasonality analysis

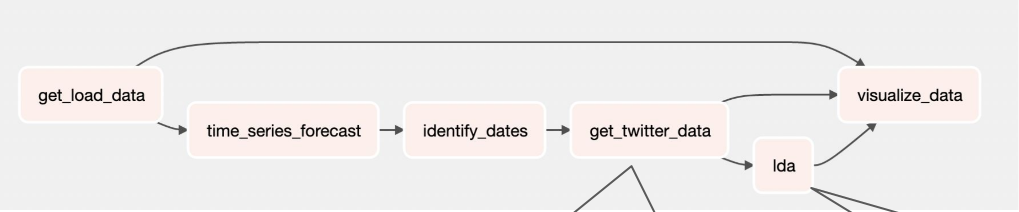



Based on RMSE

```
filtered_dates = (dates_df[(dates_df['y_abs_diff']>(5*dates_df['rmse']))]
                  .groupby(['zone_name', dates_df['ds'].dt.date])
                  .agg(anomaly_occurence=('ds', 'nunique'))
                  .reset_index())
filtered_dates = filtered_dates[filtered_dates['anomaly_occurence']>3]
filtered_dates#.describe()
```

| | zone_name | ds | anomaly_occurence |
|---|---|---|---|
| **7** | CAPITL | 2011-07-21 | 4 |
| **8** | CAPITL | 2011-07-22 | 5 |
| **14** | CAPITL | 2012-06-20 | 5 |
| **15** | CAPITL | 2012-06-21 | 6 |
| **46** | CAPITL | 2013-07-18 | 5 |
| **...** | ... | ... | ... |
| **1954** | WEST | 2013-09-11 | 6 |
| **2028** | WEST | 2016-08-12 | 4 |
| **2071** | WEST | 2019-06-28 | 4 |
| **2073** | WEST | 2019-07-02 | 4 |
| **2120** | WEST | 2021-06-29 | 4 |

# Get Twitter Data & LDA Topic Modelling




1. Get Twitter data on the identified dates
   a. Need full archive search access of Twitter API
   b. Store tweets in Storage (unstructured data)
2. Preprocess Data
   a. Remove: links, username, hashtags, medias
   b. Vectorize: lemmatize, tokenize
3. Do topic modelling with LDA
   a. **Hyperparameters tuned:** number of topics

## Get Tweets

```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:'
                '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)
```

## 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.
```

1. Preprocess Twitter Data
   a. Remove Links
   b. Remove Usernames
   c. Remove Hashtags
   d. Lemmatization and Tokenization

```python
def run_LDA(dataframe, num_topics):
    text_dict = Dictionary(dataframe.tokens)
    tweets_bow = [text_dict.doc2bow(tweet) for tweet in dataframe['tokens']]
    k = num_topics
    tweets_lda = LdaModel(tweets_bow,
                          num_topics=k,
                          id2word=text_dict,
                          random_state=2,
                          passes=10,
                          iterations=100)

    return tweets_lda, tweets_bow
```

# RESULTS

# Results - Topics that Influenced Electricity Use

## Traffic / Accidents

- Topic of **traffic includes accidents** caused by weather, congestion, etc.
- Traffic topic mostly showed up in **car-heavy city regions of NY state**, like Albany & Buffalo
- NYC had nothing with traffic which was interesting, possibly due to prevalence of public transportation

- Related to energy use because accidents cause increase **strain on emergency resources like police/hospitals** which leads to **increased energy use**
- Traffic congestion causes people to **work later, buildings open longer**

## Road / Regular Construction

- Also, **mostly present in larger cities**
- Could be **related to traffic/accidents** because construction work **congests roads**
- Could also **cause accidents**

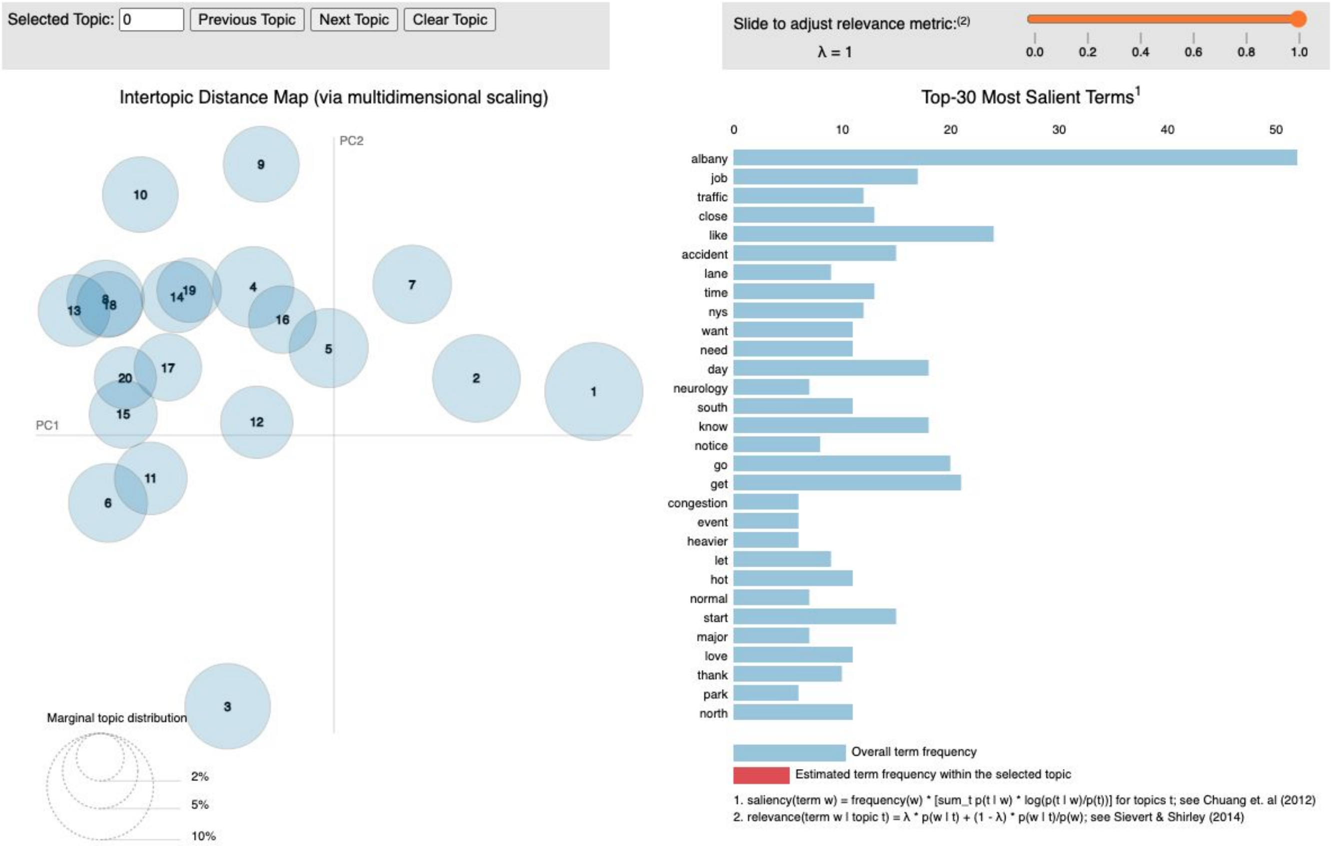
- Road and regular construction naturally is **electricity intensive**: electric equipment/machinery
- **Accidents caused by construction** will affect energy use due to above reasons

## Cultural Events

- NYC had **irrelevant topics related to NYC tourism**, like pictures, monuments, etc.
- **Had to filter** the above to show relevant topics
- Most relevant topics included cultural events like **sports games, and Fashion Week (NYFW)**

- Sporting events cause more people to watch games on TV, go to sports bars; **stadiums are massive electric loads**
- NYFW was surprising, however it brings in **added tourism, media, so more electricity use**

# Interesting Topics from LDA



# Results - Topics that Influenced Electricity Use

## Traffic / Accidents

- Topic of **traffic includes accidents** caused by weather, congestion, etc.
- Traffic topic mostly showed up in **car-heavy city regions of NY state**, like Albany & Buffalo
- NYC had nothing with traffic which was interesting, possibly due to prevalence of public transportation

- Related to energy use because accidents cause increase **strain on emergency resources like police/hospitals** which leads to **increased energy use**
- Traffic congestion causes people to **work later, buildings open longer**

## **Road / Regular Construction**

- Also, **mostly present in larger cities**
- Could be **related to traffic/accidents** because construction work **congests roads**
- Could also **cause accidents**

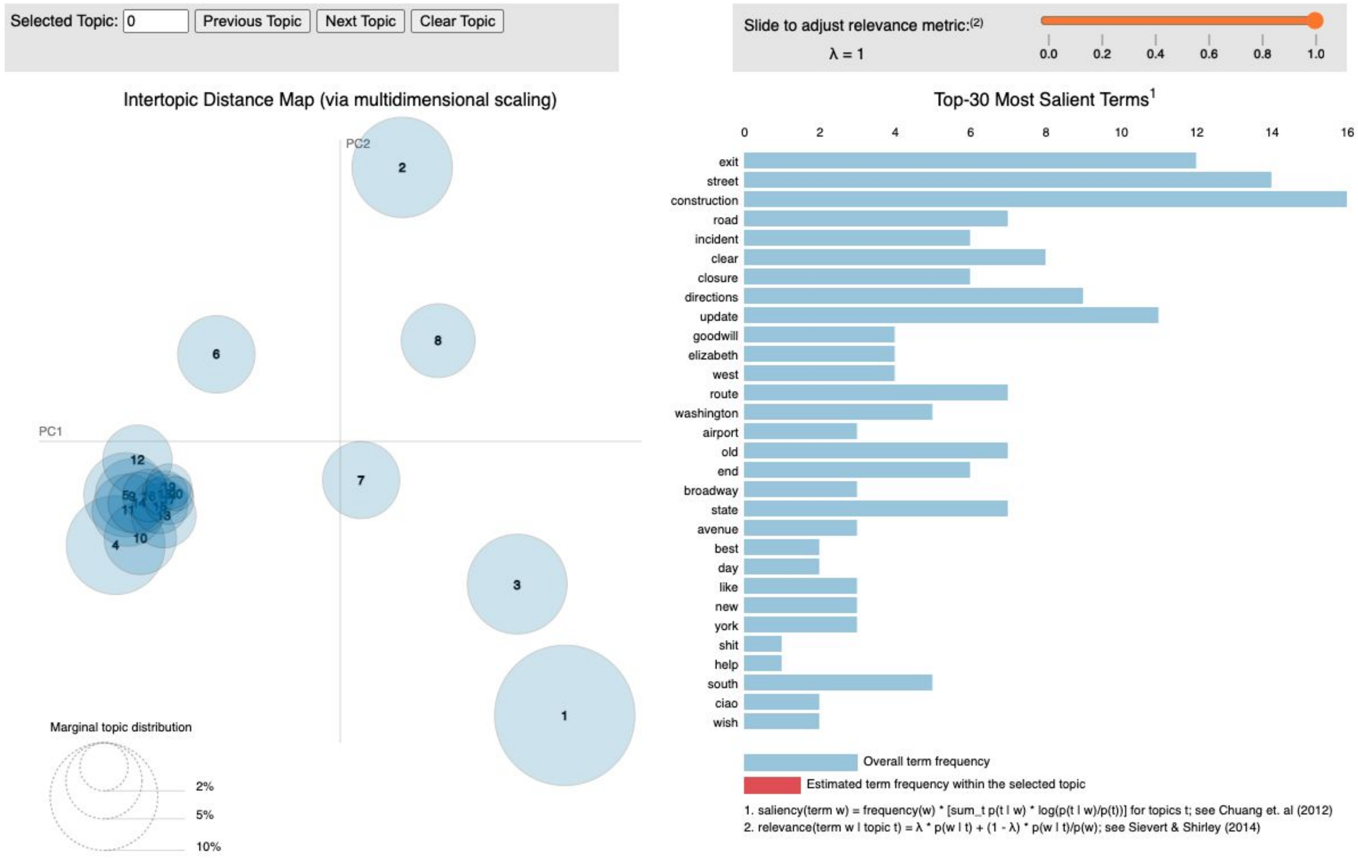
- Road and regular construction naturally is **electricity intensive**: electric equipment/machinery
- **Accidents caused by construction** will affect energy use due to above reasons

## Cultural Events

- NYC had **irrelevant topics related to NYC tourism**, like pictures, monuments, etc.
- **Had to filter** the above to show relevant topics
- Most relevant topics included cultural events like **sports games, and Fashion Week (NYFW)**

- Sporting events cause more people to watch games on TV, go to sports bars; **stadiums are massive electric loads**
- NYFW was surprising, however it brings in **added tourism, media, so more electricity use**

# Interesting Topics from LDA



# Results - Topics that Influenced Electricity Use

## Traffic / Accidents

- Topic of **traffic includes accidents** caused by weather, congestion, etc.
- Traffic topic mostly showed up in **car-heavy city regions of NY state**, like Albany & Buffalo
- NYC had nothing with traffic which was interesting, possibly due to prevalence of public transportation

- Related to energy use because accidents cause increase **strain on emergency resources like police/hospitals** which leads to **increased energy use**
- Traffic congestion causes people to **work later, buildings open longer**

## Road / Regular Construction

- Also, **mostly present in larger cities**
- Could be **related to traffic/accidents** because construction work **congests roads**
- Could also **cause accidents**

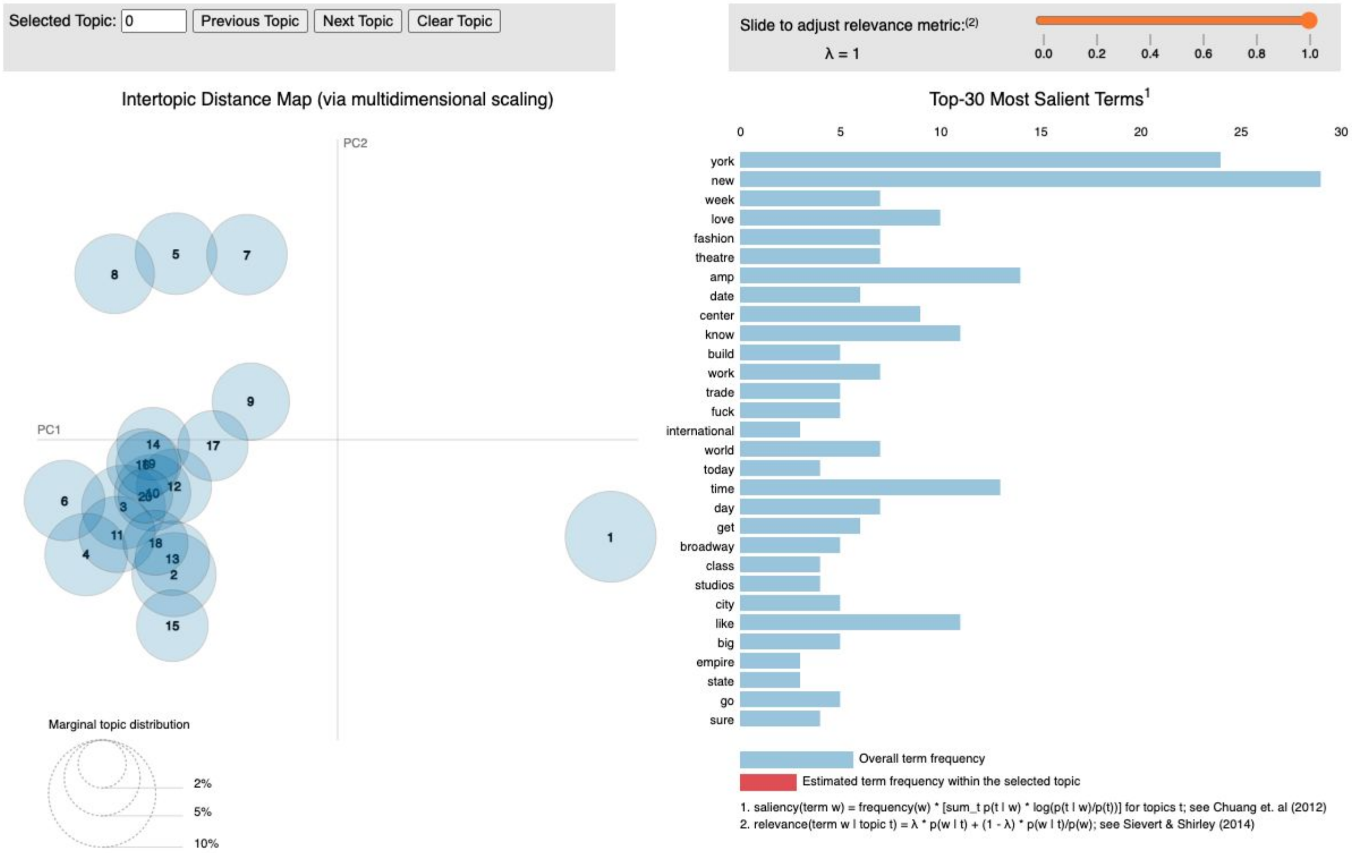
- Road and regular construction naturally is **electricity intensive**: electric equipment/machinery
- **Accidents caused by construction** will affect energy use due to above reasons

## Cultural Events

- NYC had **irrelevant topics related to NYC tourism**, like pictures, monuments, etc.
- **Had to filter** the above to show relevant topics
- Most relevant topics included cultural events like **sports games, and Fashion Week (NYFW)**

- Sporting events cause more people to watch games on TV, go to sports bars; **stadiums are massive electric loads**
- NYFW was surprising, however it brings in **added tourism, media, so more electricity use**

# Interesting Topics from LDA



# Business Value

- **Efficient prediction of anomalous spikes in energy demand is invaluable to utility companies.**
  - Appropriate resource allocation and pricing
  - Fewer losses to be incurred during unexpected spikes in demand.
- **Our analysis has found that utilizing social media to monitor certain topics can be used to predict energy demand anomalies.**
  - Of the topics we found, utility companies only monitor weather.
  - Our analysis has found that traffic incidents and large cultural events are highly correlated with energy demand spikes.
- *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*

# Demo

1. Filters:
   a. Aggregation: hourly, daily, weekly, monthly, yearly
   b. Start and end date
   c. Area to be observed
2. Visualization
   a. Electricity load energy demand line chart
   b. LDA topics visualization
      i. Bubble chart for intertopic distance
      ii. Bar chart for viewing top terms in the each topic with term frequency sorted by saliency and relevance