2016 CampusLab

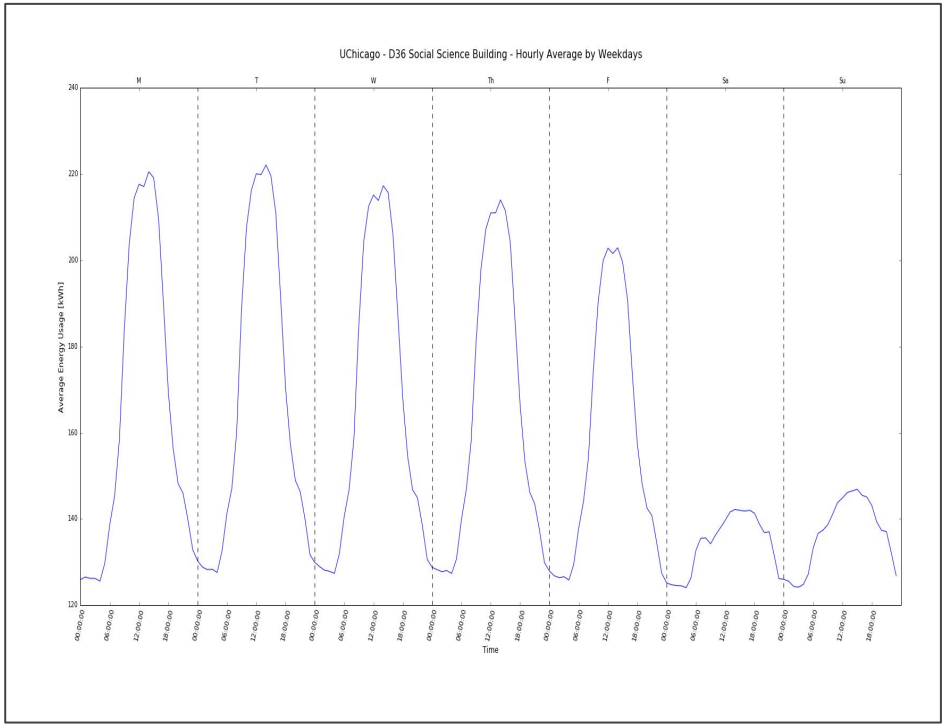
Detecting Unusual Trends in Electricity Usage Data

CS with Applications III (CS 12300 / CAPP 30123)
The University of Chicago, Spring 2016
Christine Chung and Leith McIndewar

University of Chicago Energy Data

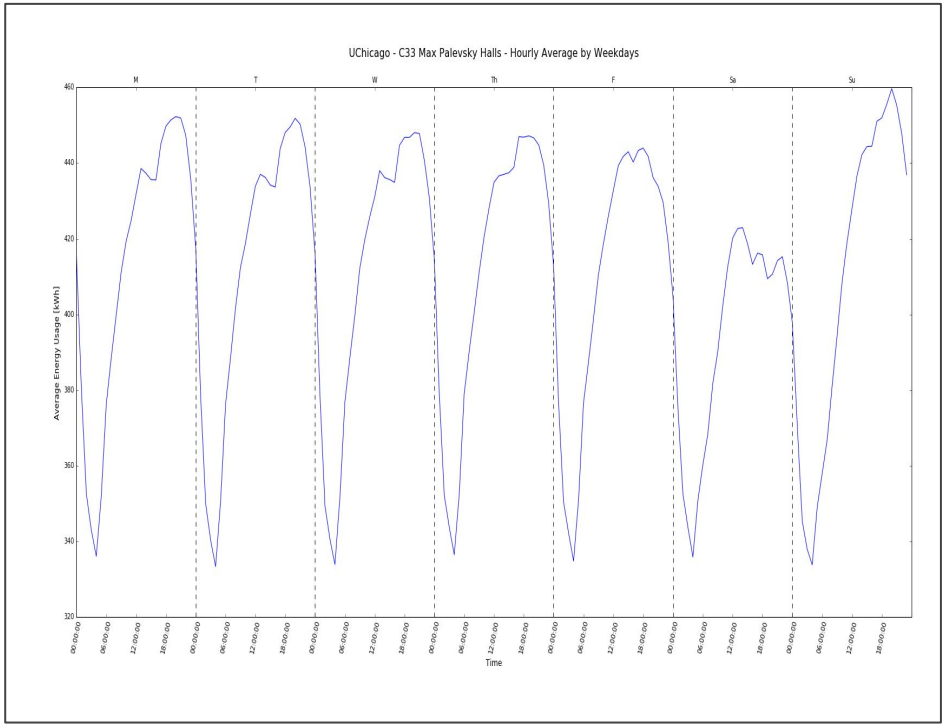
Data

- UChicago is one of the largest energy consumers in Chicago (with O'Hare)
- ~120 buildings
- Over 300 meters
- Meters and buildings do not always have a clear relationships



University of Chicago Energy Data

- Right now data is available at half hour intervals, but moving to more granular readings
- Purchase power hourly
- How would we detect if usage in a building is abnormal?
- **Goals:** Pattern & Anomaly Detection



Tools and Implementation

Tools

- PostgreSQL
- Amazon EC2, EMR, and S3
- ElasticSearch & Kibana
- MRJob
- Python Threading and Queue Class

Algorithms

For anomaly detection:

- Tarzan (Suffix Trees, Markov Models)
- K-Nearest Neighbors

CAAL - Dashboard - Kibana - Google Chrome

CAAL - Dashboard

campuslab.rdcep.org/app/kibana#/dashboard/CAAL?_g=(refreshInterval:(display:Off,pause:!f,value:0),time:(from:'2014-02-20T13:33:58.786Z',mode:absolute,to:'2016-03-24T01:45:52.836Z'))&_a=(

kibana | Discover | Visualize | Dashboard | Settings | February 20th 2014, 07:33:58.786 to March 23rd 2016, 20:45:52.836

CAAL | *

Measurements Average Usage

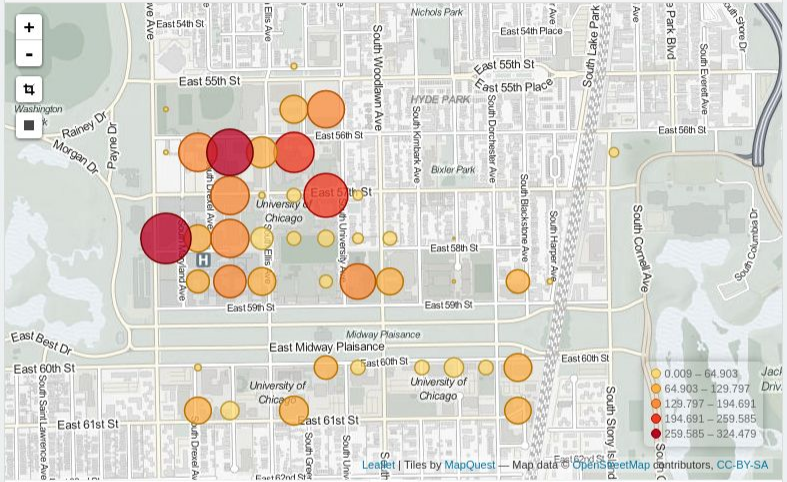
5,954,195

Count

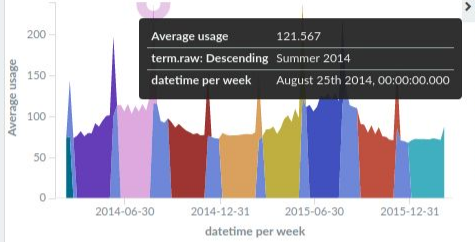
91.341

Average usage

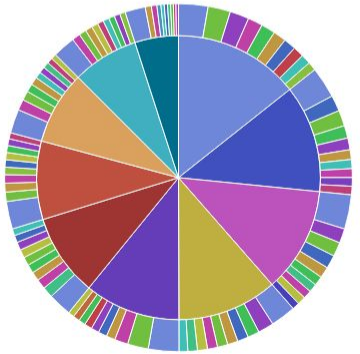
Building usage distribution



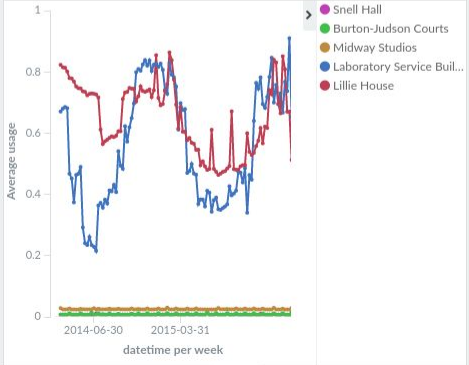
Usage per Term



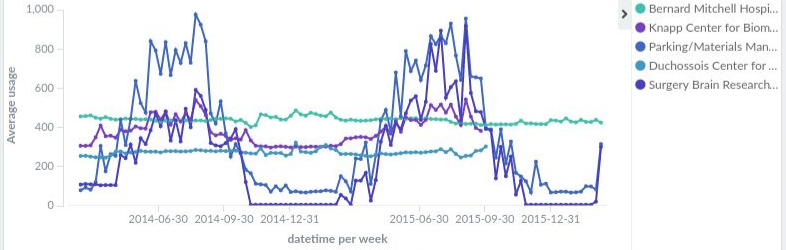
Usage by term and building



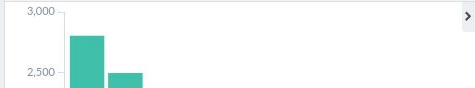
Less demanding buildings



Most demanding buildings



Usage bar chart per building



Data Flow

AWS EC2 with Postgres

AWS EMR with ElasticSearch

MRJob

K-NN

Vectorization

Tarzan

Discretization

MRJob

Multithreading

Pandas

Algorithms: Tarzan

Tarzan Algorithm

Finding Surprising Patterns in a Time Series Database in Linear Time and Space – Keogh et al. *SIGKDD 2002*

Step 1:

Discretize time series into strings

Step 2:

Preprocess strings with Suffix Trees

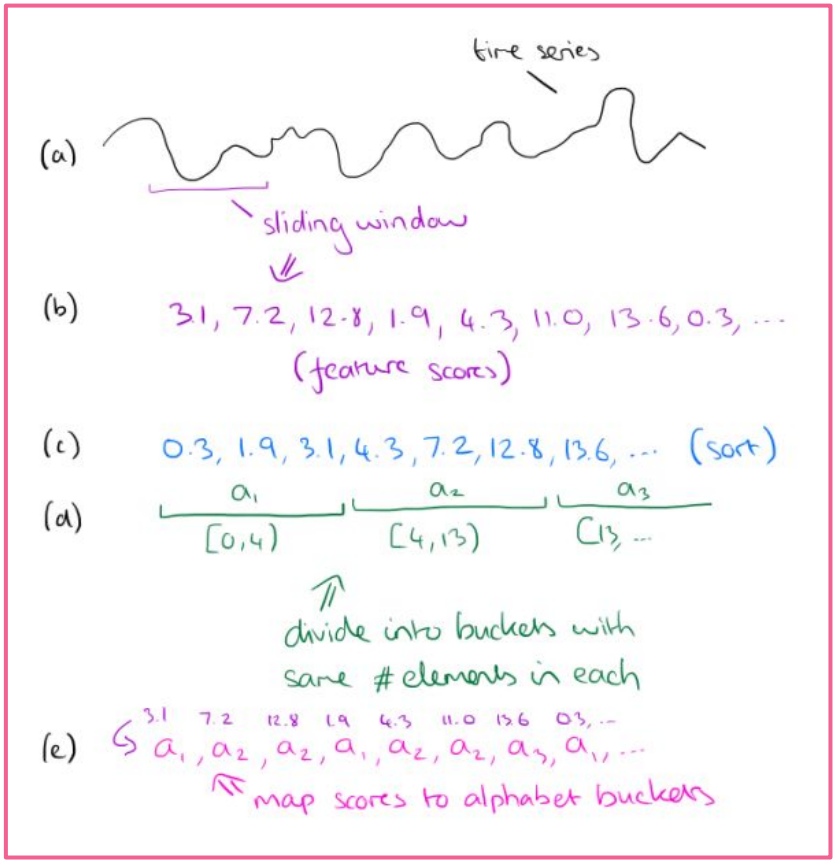
Step 3:

Score strings with Markov Models

```
void TARZAN (time_series R, time_series X,
             int l1, int a, int l2, real c)
    let x = DISCRETIZE_TIME_SERIES (X, l1, a)
    let r = DISCRETIZE_TIME_SERIES (R, l1, a)
    let Tx = PREPROCESS (r, x)
    for i = 1, |x| − l2 + 1
        let w = x[i,i+l2−1]
        retrieve z(w) from Tx
        if |z(w)| > c then print i, z(w)
```

Table 4: Outline of the Tarzan algoritm: l_1 is the feature window length, a is the alphabet size for the discretization, l_2 is the scanning window length and c is the threshold

source: http://www.cs.ucr.edu/~eamonn/sigkdd_tarzan.pdf

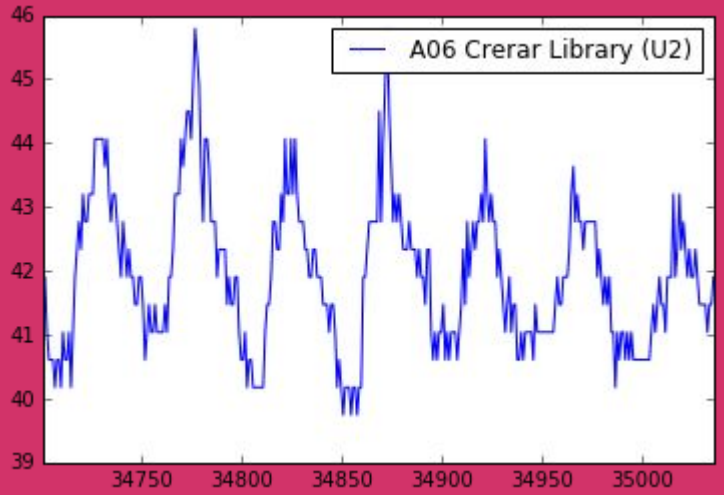


Discretization

1. Slide window over feature scores.
2. Generate one real number (slope)
3. Bucket slopes by quantiles
4. Assign symbol to each bucket
5. Replace numbers in original time series with label from bucket

Source: https://blog.acolyer.org/2016/05/09/finding-surprising-patterns-in-a-time-series-database-in-linear-time-and-space/

Step 1: time series -----> string



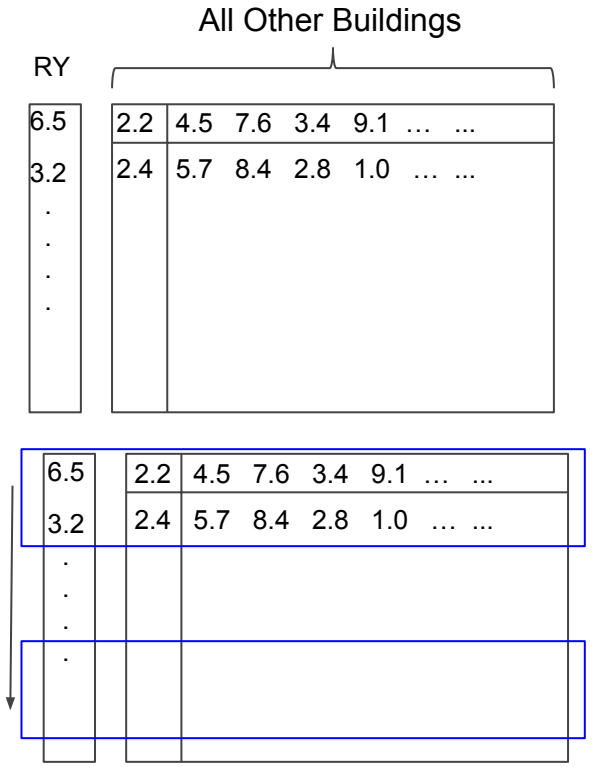
=

edaafmhkkkjgjklmonkkippooohhafdeelib
bgkgccebahphdejneileahplnnnmmoonn
oojkllgfggkkdeccdgphcddfhpfedjggfhba
hhhommpmmlalnljjhjkgfeaaabaipjabaab
aelidedccmpbfmjfmlkloophhkjjjkjggfdob
bbapleabfcebkngffcnmoggdnmpafnmlli
kjoomplljgebedddcbcnjggfcfmnmphakk
kahhhhhhpoofhpppmmkgdebfmphhhcd
dccbnddeggnmhfcnceahhhhhhpononof
bknnljhjljdccceilcbbhaelp

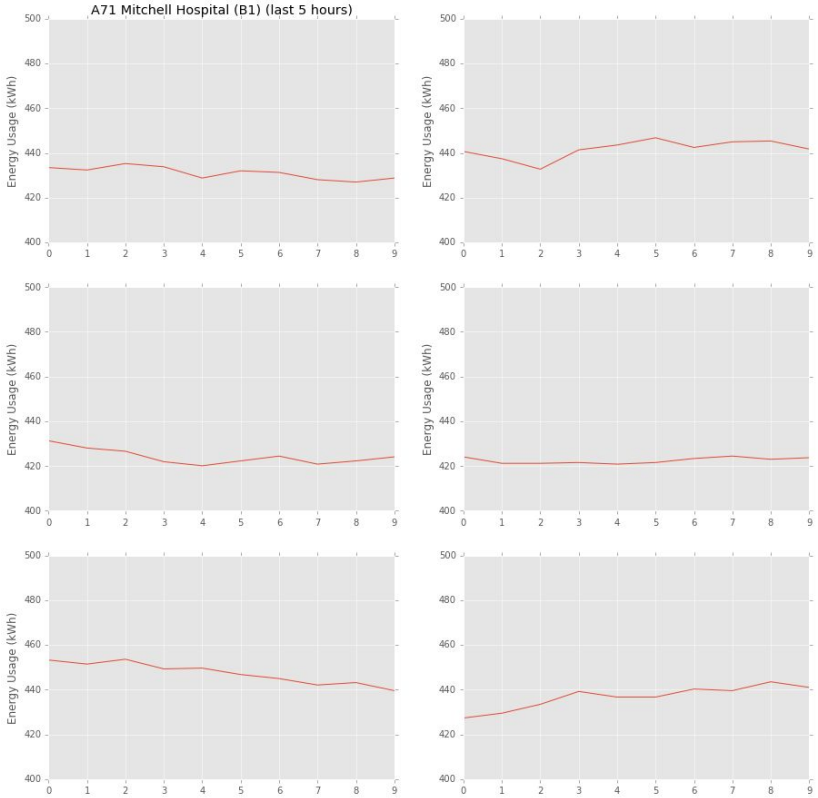
Algorithms: K-Nearest Neighbors

K Nearest Neighbors

- Define a window for comparison and k parameter
- For each building, compare the readings from all other buildings to find the k nearest neighbors
- Find what that building is doing during its ‘nearest neighbors’
- Calculate distance to between those readings and the current readings
- Repeat for each building



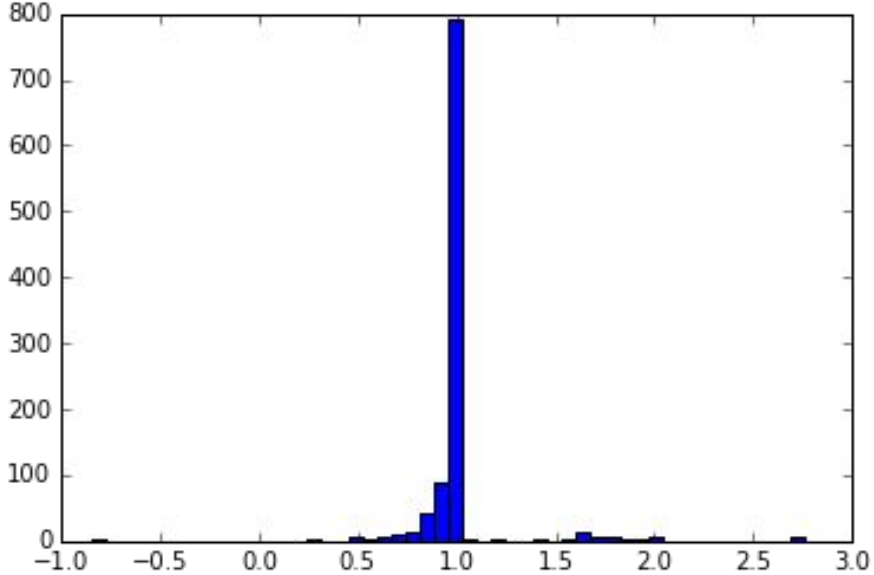
K Nearest Neighbors

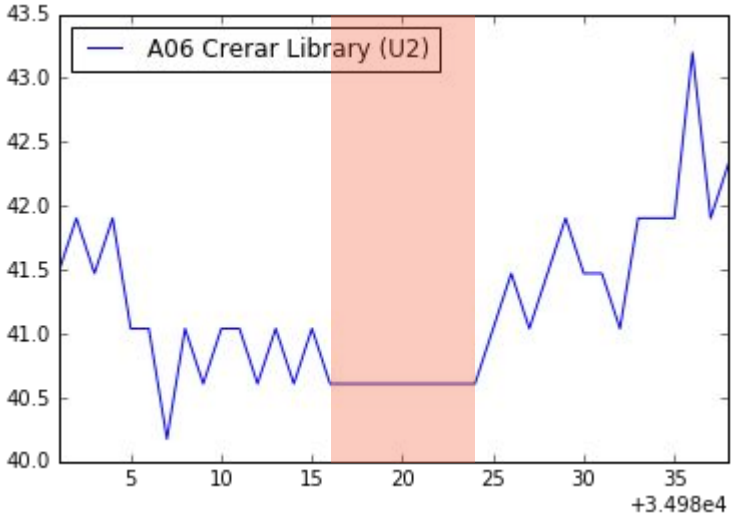


- Nearest Neighbors for Mitchell Hospital meter
- Distance between current readings (upper left) and average of 5 nearest neighbors:
  - 14.146

Results

Distribution of “Surprise Scores”





	datetime	A06 Crerar Library (U2)
34995	2016-04-10 03:00:00	41.040
34996	2016-04-10 03:30:00	40.608
34997	2016-04-10 04:00:00	40.608
34998	2016-04-10 04:30:00	40.608
34999	2016-04-10 05:00:00	40.608
35000	2016-04-10 05:30:00	40.608
35001	2016-04-10 06:00:00	40.608
35002	2016-04-10 06:30:00	40.608
35003	2016-04-10 07:00:00	40.608
35004	2016-04-10 07:30:00	40.608

“Surprise Result”

April 10, 2016 | Crerar Library

4 hours of unchanging usage from 3:30am - 7:30am

Run Times

K-NN	~ 31 min
K-NN with MR Job	~ 26 min
Tarzan	~ 17 min
Tarzan w/ Multithreading	~ ∞ min

Next Steps

Sharding with EMR and ElasticSearch

Effectively use MR Job and Multithreading with both Algorithms

Reduce Run Times

Questions?