

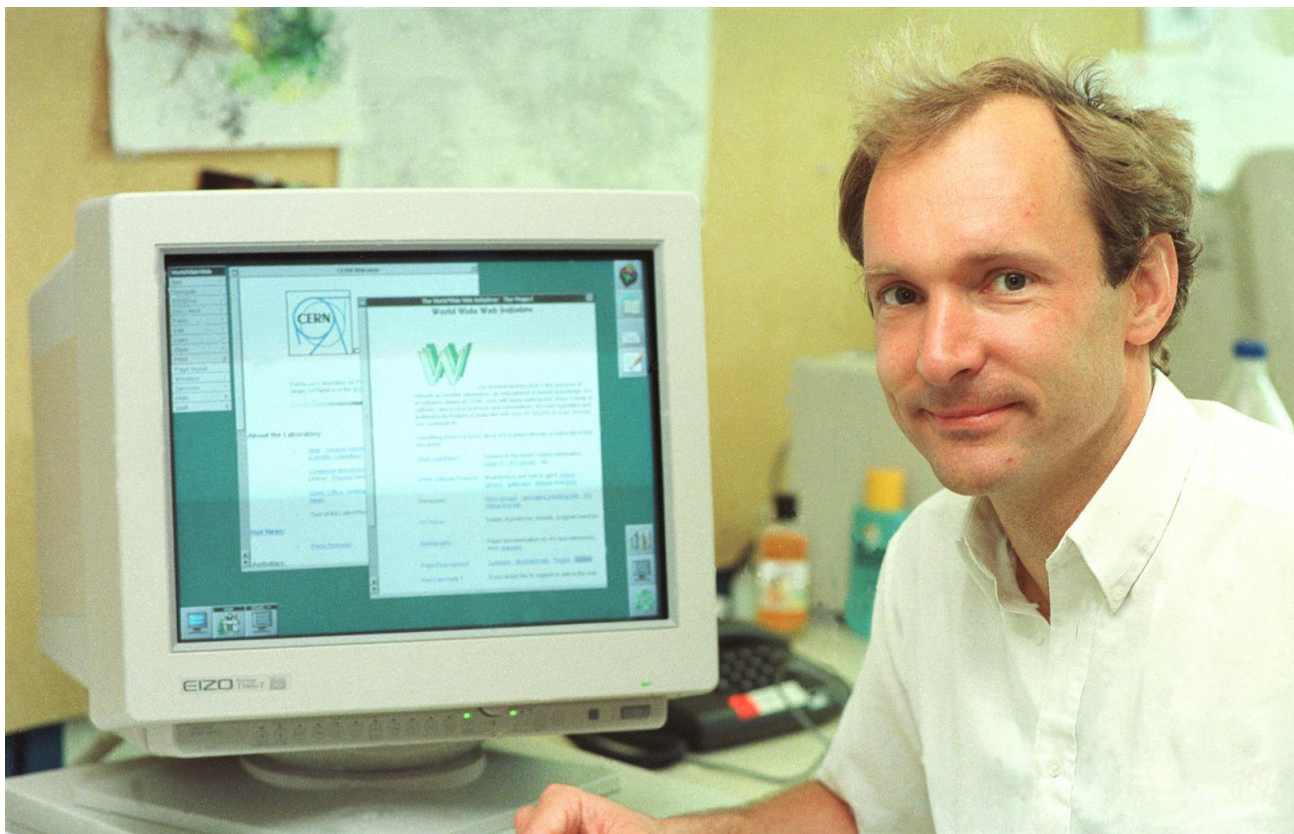
Real-World Python

Brahm Capoor

Download today's lecture code [here](#), complete with some dummy data (instructions for downloading your own facebook data can be found [here](#)).

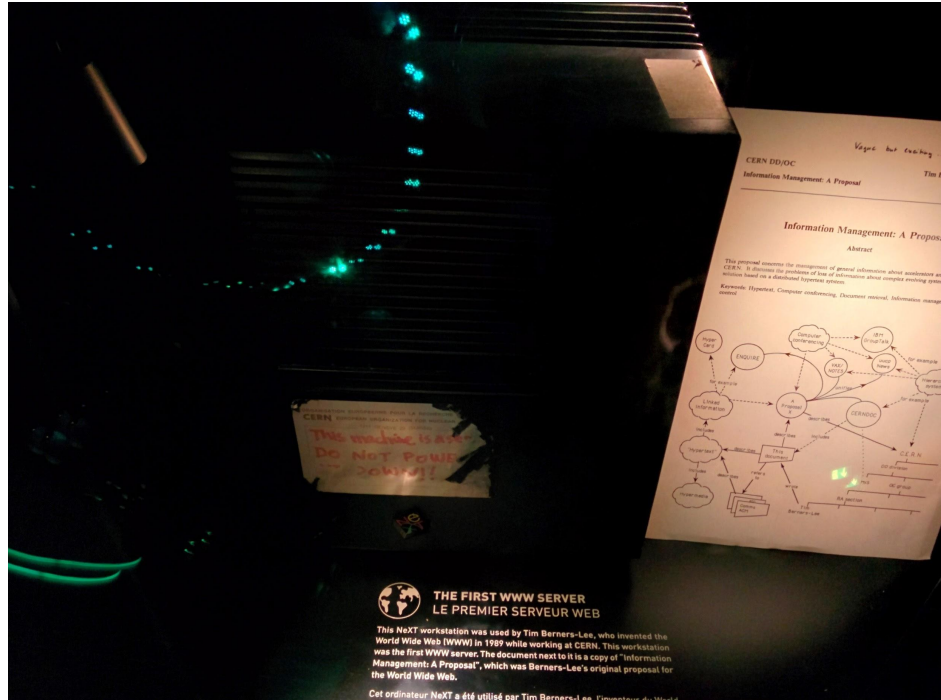
A few logistics

- Nick doesn't have office hours today
 - ... but I'll do them instead! They'll be in my office, Gates B02
- The Monday after Thanksgiving is the last class of the quarter 🙄
- I'll be holding an exam review session on **Tuesday, the 3rd of December from 7:00 to 8:30** in **Hewlett 200**
- Bluescreen project submissions are due **next Wednesday**
- We've posted some final exam review resources [here](#)









Overheard at Stanford

Open Seas group

Arrbout

Main Deck

Eric's post

Crew

Grog Fests

Portraits

Watch party

Scour the crew!!

Shortcuts

Math... be moarr than 2020

Elitist... be moarr than 2020

Board Gamers

Overheard at Stanford

Wild ... be moarr than 2020

Stanford Memes for ...

ideas

facebook.com

Brahm Home Port Hoist

Ok so it's very important to me that I'm in a power couple

Euler has his own number, Euler's equation, Euler-Cauchy, Euler-Schwarz, he's the Amalgam of math

Does FMOTQ stand for First Monday of the

Boarded Hail-shots Blabber t' yer mates More

Eric Kim

October 17

OH in Week 4

A: At some point I should go to lecture for 106A

B: Who's teaching it?

A: I don't know

Ye, Jennie Yang, Alice Yang an' 130 other scallywags 22 Scrawlin's

Yo ho ho! Yer thoughts Split the booty

Eric Kimgot em

Arrr! · Return fire · 4 o' sevens

Sathya Edamadaka Sarah MacHarg..... we know who A is

Arrr! · Return fire · 4 o' sevens · Amended

Sarah MacHarg be retaliatin' · 2 Retaliations

Jennie YangBrahm is this you???

Arrr! · Return fire · 4 o' sevens

Brahm Capoor Jennie Yang wait 106A is offered this quarter?

Arrr! · Return fire · 4 o' sevens

INVITE MEMBERS

+ e-bottle

CREWMATES 4,273 Crewmates

GATHER YE'SELF A NEW CREW

Make it easier to share stories wit' ye hearties, mates, an' other scallywags.

RECENT CREW PORTRAITS

CS 224W

Suggested Crews

Peter Maldonado

OCTOBARR 30, 2019, 9:54 IN THE MORNIN'

Peer into

Peer into

type a message...

CREW COLLOQUIES

MORE CONTACTS (126)

Search

facebook.com

Overheard at Stanford

Brahm Home Port Hoist

Overheard at Stanford

Open Seas group

Arrbout

Main Deck

Eric's post

Crew

Grog Fests

Portraits

Watch party

Scour the crew!!

Shortcuts

Math... be moarr than 2020

Elitist... be moarr than 2020

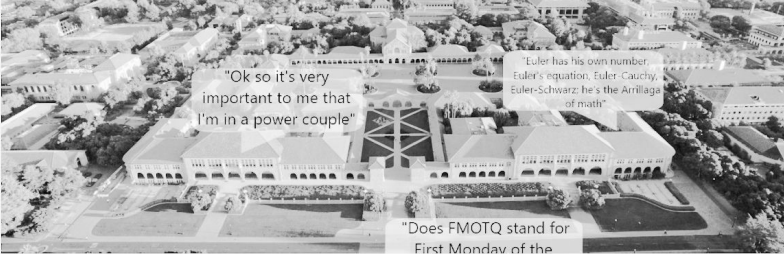
Board Gamers

Overheard at Stanford

Wild ... be moarr than 2020

Stanford Memes for ... 8

ideas 7



"Ok so it's very important to me that I'm in a power couple"

Euler has his own number. Euler's equation. Euler-Cauchy. Euler-Schwarz he's the Arrbagg of math"

"Does FMOTQ stand for First Mondav of the

Boarded

Hail-shots

Blabber t' yer mates

More

Eric Kim

October 17

OH in Week 4

A: At some point I should go to lecture for 106A

B: Who's teaching it?

A: I don't know

Ye, Jennie Yang, Alice Yang an' 130 other scallywags 22 Scrawlin's

Yo ho ho!

Yer thoughts

Split the booty

Eric Kingot em 2

Arrr! - Return fire - 4 o' sevens

Sathya Edamadaka Sarah MacHarg..... we know who A is

Arrr! - Return fire - 4 o' sevens - Amended

Sarah MacHarg be retaliatin' - 2 Retaliations

Jennie YangBrahm is this you???

Arrr! - Return fire - 4 o' sevens

Brahm Capoor Jennie Yang wait 106A is offered this quarter?

Arrr! - Return fire - 4 o' sevens

INVITE MEMBERS

+ e-bottle

CREWMATES 4,273 Crewmates

GATHER YE'SELF A NEW CREW

Make it easier to share stories wit' ye hearties, matesys, an' other scallywags.

RECENT CREW PORTRAITS

Peer into

Suggested Crews

Peer into

Peter Maldonado

OCTOBARR 30, 2019, 9:54 IN THE MORNIN'

More contacts (126)

In Wednesday's class, we introduced the `urllib` module, which allows us to `request` information like text or HTML files from the internet like so:

```
>>> f = urllib.request.urlopen('http://web.stanford.edu/class/cs106a/hello.txt')
>>> f.read().decode('utf-8')

'Hello from CS106A url!\nWhat if there were data here?\n12,34\n25,19\n66,0\n1,2\n'
```

In Wednesday's class, we introduced the `urllib` module, which allows us to `request` information like text or HTML files from the internet like so:

```
>>> f = urllib.request.urlopen('http://web.stanford.edu/class/cs106a/hello.txt')
>>> f.read().decode('utf-8')
'Hello from CS106A url!\nWhat if there were data here?\n12,34\n25,19\n66,0\n1,2\n'
```

This is only one piece of the puzzle.

Even the world's simplest chat application must have two kinds of request:

- **Getting** all the messages from a server
- **Sending** a message to a server

and any server worth its salt should gracefully deal with both requests somehow

Even the world's simplest chat application must have two kinds of request:

- Getting all the messages from a server
- **Sending** a message to a server

and any server worth its salt should gracefully deal with both requests somehow

If you'd like to try the chat app, download the lecture code [here](#).

The online logfile for the app can be found [here](#)

Now, we have some sense of the kinds of things that a chat application needs to do in order to function.

More importantly, we've seen that servers can **store and log** all of the messages that are sent, and it shouldn't be too hard to imagine that we could have a better log.

Now, we have some sense of the kinds of things that a chat application needs to do in order to function.

More importantly, we've seen that servers can store and log all of the messages that are sent, and it shouldn't be too hard to imagine that we could have a better log.

Let's do something cool.

Download your information

You can download a copy of your Facebook information at any time. You can download all of it at once, or you can select only the types of information and date ranges that you want. You can choose to receive your information in an HTML format that is easy to view, or a JSON format, which could allow another service to more easily import it.

Downloading your information is a password-protected process that only you will have access to. Once you've created a file, it will be available for download for a few days.

If you'd like to view your information without downloading it, you can [Access your information](#) at any time.

New file

Available files ¹

Date range:

All of my data ▾

Format:

JSON ▾

Media quality:

Medium ▾

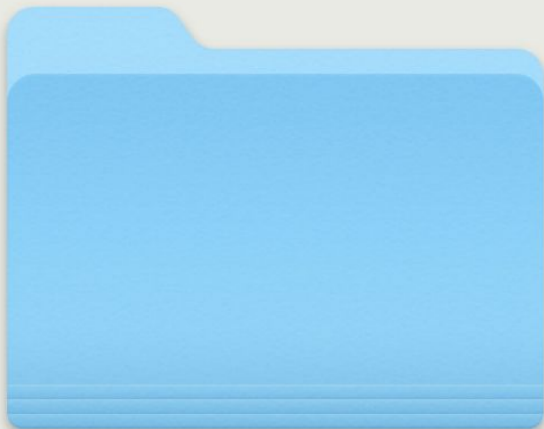
Create File

Your information ¹

Select all

	Posts Posts you've shared on Facebook, posts that are hidden from your timeline and polls you have created	☐
	Photos and videos Photos and videos that you've uploaded and shared	☐
	Comments Comments you've posted on your own posts, on other people's posts or in groups that you belong to	☐
	Likes and reactions Posts, comments and Pages that you've liked or reacted to	☐
	Me Hearties The people you are connected to on Facebook	☐
	Stories Photos and videos you've shared to your story	☐
	Following and followers People, organisations or business that you choose to see content from, and people who follow you	☐
	Messages Messages you've exchanged with other people on Messenger	☒

facebook.com/dyi



messages

3.09 GB, 5 items

Last modified on Nov 20, 2019 at 10:45:14 AM

The first facebook message I ever received:

“ur profile pic is lame and u have one friend. LOL!! btw, facebook is going to be useless to u”

The first facebook message I ever received:

“ur profile pic is lame and u have one friend. LOL!! btw, facebook is going to be useless to u”

Since then, I have sent or received more than 350,000 messages.

Each line of the file looks like this:

`<timestamp> <friend> <message> <status>`

`<timestamp>`: how many months after my account's creation the message was sent

`<friend>`: my conversation partner

`<message>`: the body of the message

`<status>`: Whether I `received` or `sent` the message.

These fields are separated by tab characters (`'\t'`)

Getting your own data

- You can download your own data from facebook, but unfortunately, it doesn't come in the format we're using today
 - It's a format that's useful for many kinds of programs, but a little too complex for our purposes
- I wrote a Python Program that takes the files you can download from facebook, and converts them to the format I described in the previous slide
- If you're interested in making your own data file, follow the instructions in the next few slides. Otherwise, [skip forward to the main lecture](#).

Getting your own data

- First, request to [download your own data](#) from facebook.
 - Click 'download your information'
 - In the top bar, select '**JSON**' from the 'Format' dropdown
 - Click 'Deselect All' at the top right of the options menu, and then scroll down and reselect 'Messages'
 - You can download whatever else you want (it's all super interesting!), but the download will take longer to process. I don't recommend downloading your photos and videos, because the size of the download increases almost exponentially.
 - Click 'Create File'
 - Facebook will take a while to create the file, but will shoot you an email within about an hour (if you're downloading only your messages) with a link to the download
 - As a side note, it feels really cool to be doing something that takes even a company the size of facebook a while :)

Getting your own data

- Once you've downloaded your data, unzip it and open it up.
- You should see a folder called **messages/** which contains at least some of the following subfolders:
 - **archived_threads/**
 - **filtered_threads/**
 - **inbox/**
 - **message_requests/**
 - **stickers_used/**
- Open the **inbox/** folder, and download and copy [this Python program](#) into it. You'll need to right click the link on this slide and click 'save as'.

Getting your own data

- Open `parse_messages.py`
- You are *not* responsible for understanding this code, but it is exhaustively commented if you're interested in reading it!
- Modify the `MY_NAME`, `ACCOUNT_CREATION_YEAR` and `ACCOUNT_CREATION_MONTH` constants to be your details.
- Open the `inbox/` folder in your terminal, and run this command (it'll take a while):

```
python3 parse_messages.py
```

- I've only tested this script on my own data, email me if anything goes wrong.

Getting your own data

- If `parse_messages.py` completed successfully, a new file should have been created in the `inbox/` folder called `message_logs.txt`
- Copy `message_logs.txt` into this lecture's code folder, and you're ready to start analyzing! `chat_utils.py` and `Chat Analysis.ipynb` should already be set up to work with it.

We're going to ask two questions of our data today:

- 1) Who do I talk to, and how much do I talk to them?
- 2) What do I talk about?

We're going to ask two questions of our data today:

- 1) Who do I talk to, and how much do I talk to them?
- 2) What do I talk about?

To answer them, we're going to calculate and store two kinds of data:

- 1) How often I talk to **each of my friends**
- 2) How often I use **particular words**

How often do I talk to my friends?

I want to know — for **each friend** I've sent a message to — **how many** messages I've sent them **per month** that I have messaged them.

How often do I talk to my friends?

I want to know — for each friend I've sent a message to — how many messages I've sent them per month that I have messaged them.

I want to associate my friend's **names** with **dates** (in this case, months), and subsequently with chat **counts** (which we can also call ranks)

How often do I talk to my friends?

I want to know — for each friend I've sent a message to — how many messages I've sent them per month that I have messaged them.

I want to associate my friend's names with dates (in this case, months), and subsequently with chat counts (which we can also call ranks)

Sound familiar?

How often do I talk to my friends?

I want to know — for each friend I've sent a message to — how many messages I've sent them per month that I have messaged them.

I want to associate my friend's names with dates (in this case, months), and subsequently with chat counts (which we can also call ranks)

Sound familiar? It's basically BabyNames!

```
def add_one_to_count(counts, counted, month):
    if counted not in counts:
        counts[counted] = {}

    inner_counts = counts[counted]
    if month not in inner_counts:
        inner_counts[month] = 0

    inner_counts[month] += 1

    return counts

def get_counts_per_person():
    counts = {} # initialize a counts dict
    with open(DATA_FILENAME, 'r') as f:
        for line in f:
            # turn a line into a list of constituent parts,
            # and yank out the relevant bit
            line_parts = line.split('\t')
            month = int(line_parts[0])
            conversation_partner = line_parts[1]

            counts = add_one_to_count(counts, conversation_partner, month)

    return counts
```

How often do I use particular words?

I want to know — for each word that I've used in a message — how many messages per month have used that word.

How often do I use particular words?

I want to know — for each word that I've used in a message — how many messages per month have used that word.

I want to associate **each word** with **dates** (in this case, months) and subsequently with **frequencies**

How often do I use particular words?

I want to know — for each word that I've used in a message — how many messages per month have used that word.

I want to associate each word with dates (in this case, months) and subsequently with frequencies

Sound familiar?

How often do I use particular words?

I want to know — for each word that I've used in a message — how many messages per month have used that word.

I want to associate each word with dates (in this case, months) and subsequently with frequencies

Sound familiar? *It's literally exactly what we just did*

```
def get_chat_word_counts():
    counts = {} # initialize a counts dict
    with open(DATA_FILENAME, 'r') as f:
        for line in f:
            # turn a line into a list of constituent parts,
            # and yank out the relevant bit
            line_parts = line.split('\t')
            month = int(line_parts[0])
            content = without_punctuation(line_parts[2].lower())
            status = line_parts[3]

            if status == "received":
                # we're only considering messages that I sent
                # so if we find one that I received, move on
                # with life
                continue

            # return a list of individual words in the message
            words = content.split()

            for word in words:
                counts = add_one_to_count(counts, word, month)

    return counts
```

Have a wonderful break!

