Real-World Python

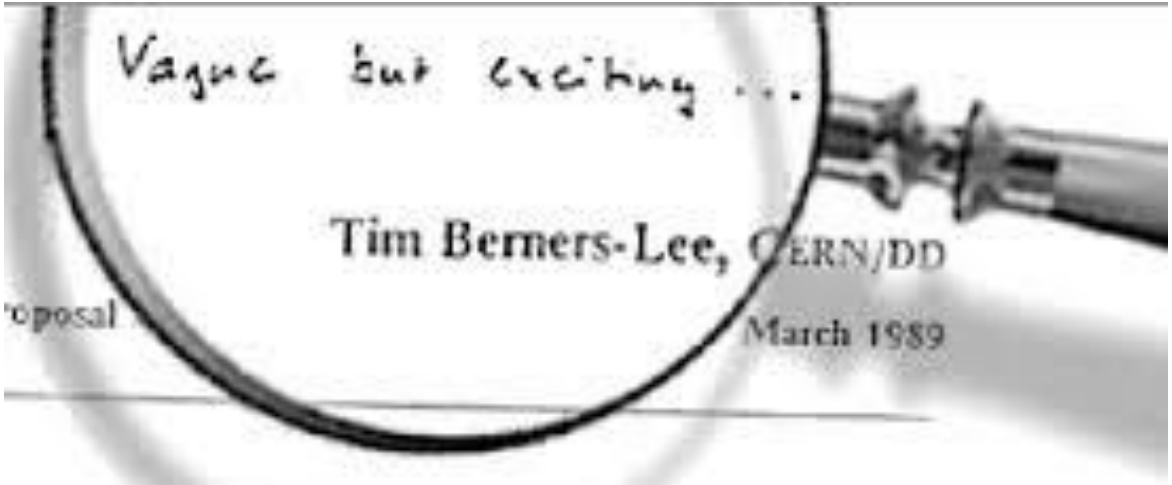
Brahm Capoor

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

- Next Monday is the last class of the quarter 😢
- I'll be holding an exam review session on Monday, the 9th of March from 7:00 to 8:30 in Hewlett 200
- We'll be posting final exam review resources early next week
- The LaIR is virtual next week (check the course page for details)
- Check out the honor code announcement we made in class on Wednesday

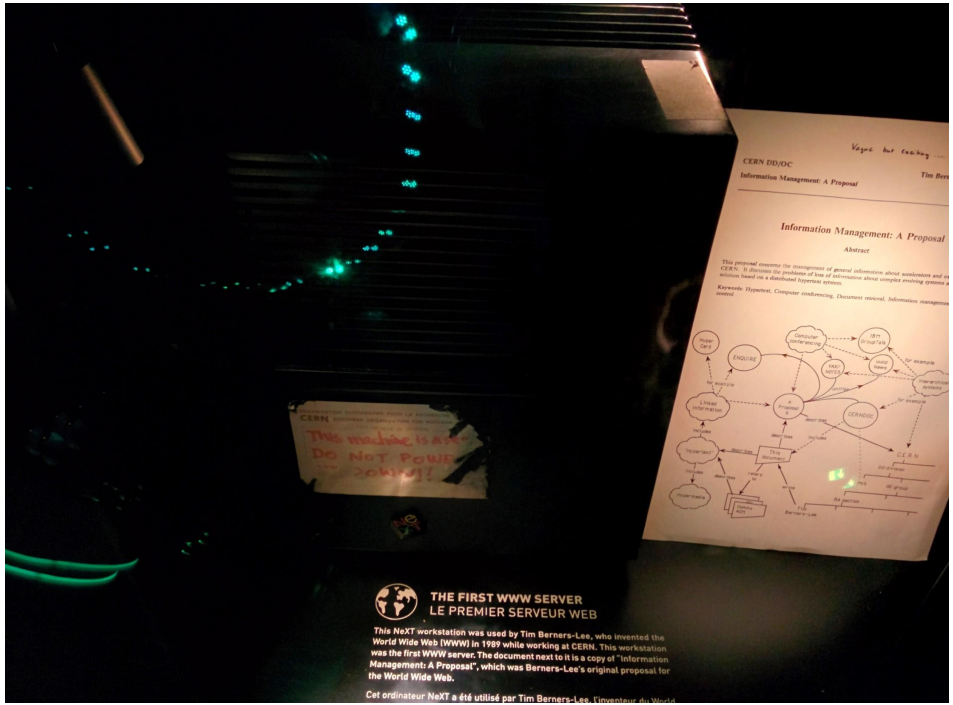

CERN
World Wide Web Initiative
EIZO

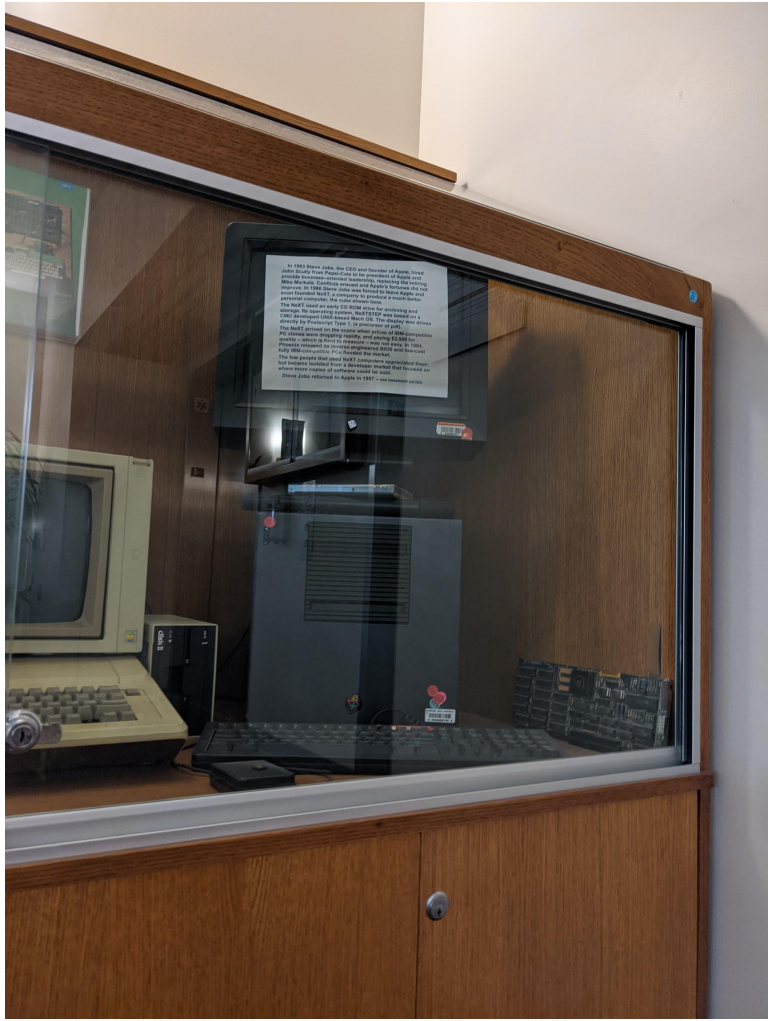


oposal

n Management: A Proposal

Abstract


CERN
CERN DD/OC
Information Management: A Proposal
Information Management: A Proposal
Abstract
THE FIRST WWW SERVER
LE PREMIER SERVEUR WEB
This NeXT workstation was used by Tim Berners-Lee, who invented the World Wide Web (WWW) in 1989 while working at CERN. This workstation was the first WWW server. The document next to it is a copy of "Information Management: A Proposal", which was Berners-Lee's original proposal for the World Wide Web.
Cet ordinateur NeXT a été utilisé par Tim Berners-Lee, l'inventeur du World


In 1983 Steve Jobs, the CEO and founder of Apple, hired
from Pepsi-Cola to be president of Apple and
leadership, replacing the retiring
. Under Sculley's fortunes did not
Steve Jobs was forced to leave Apple and
a company to produce a much better
the cube shown here.
early CD ROM drive for archiving and
. It was based on a
based Mach OS. The display was driven
(a precursor to
on the scene when prices of IBM-compatible
dropping rapidly, and paying $3,995
Macintosh – was not easy. In 1988,
reverse engineered BIOS and low-cost
PC / Macintosh
used NeXT computers and
but became isolated from a developer market that focused on
where more copies of software could be sold.
Steve Jobs returned to Apple in 1997 – see basement exhibit.

BIRTH OF THE INTERNET

THE ARCHITECTURE OF THE INTERNET AND THE DESIGN OF
THE CORE INTERNETWORKING PROTOCOL TCP (WHICH LATER BECAME TCP/IP)
WERE CONCEIVED BY VINTON G. CERF AND ROBERT E. KAHN DURING 1973
WHILE CERF WAS AT STANFORD'S DIGITAL SYSTEMS LABORATORY AND
KAHN WAS AT ARPA (LATER DARPA). IN THE SUMMER OF 1976, CERF LEFT STANFORD
TO MANAGE THE PROGRAM WITH KAHN AT ARPA.

THEIR WORK BECAME KNOWN IN SEPTEMBER 1973 AT A NETWORKING CONFERENCE IN ENGLAND.
CERF AND KAHN'S SEMINAL PAPER WAS PUBLISHED IN MAY 1974.

CERF, YOGEN K. DALAL, AND CARL SUNSHINE
WROTE THE FIRST FULL TCP SPECIFICATION IN DECEMBER 1974.
WITH THE SUPPORT OF DARPA, EARLY IMPLEMENTATIONS OF TCP (AND IP LATER)
WERE TESTED BY BOLT BERANEK AND NEWMAN (BBN),
STANFORD, AND UNIVERSITY COLLEGE LONDON DURING 1975.

BBN BUILT THE FIRST INTERNET GATEWAY, NOW KNOWN AS A ROUTER, TO LINK NETWORKS TOGETHER.
IN SUBSEQUENT YEARS, RESEARCHERS AT MIT AND USC-ISI, AMONG MANY OTHERS,
PLAYED KEY ROLES IN THE DEVELOPMENT OF THE SET OF INTERNET PROTOCOLS.

KEY STANFORD RESEARCH ASSOCIATES AND FOREIGN VISITORS

VINTON CERF

DAG BELSNES
RONALD CRANE
YOGEN DALAL
JUDITH ESTRIN
RICHARD KARP
GERALD LE LANN

JAMES MATHIS
BOB METCALFE
DARRYL RUBIN
JOHN SHOCH
CARL SUNSHINE
KUNINOBU TANNO

DARPA

ROBERT KAHN

COLLABORATING GROUPS

BOLT BERANEK AND NEWMAN
WILLIAM PLUMMER · GINNY STRAZISAR · RAY TOMLINSON

MIT
NOEL CHIAPPA · DAVID CLARK · STEPHEN KENT · DAVID P. REED

NDRE
YNGVAR LUNDH · PAAL SPILLING

UNIVERSITY COLLEGE LONDON
FRANK DEIGNAN · MARTINE GALLAND · PETER HIGGINSON
ANDREW HINCHLEY · PETER KIRSTEIN · ADRIAN STOKES

USC-ISI
ROBERT BRADEN · DANNY COHEN · DANIEL LYNCH · JON POSTEL

ULTIMATELY, THOUSANDS IF NOT TENS TO HUNDREDS OF THOUSANDS
HAVE CONTRIBUTED THEIR EXPERTISE TO THE EVOLUTION OF THE INTERNET.

DEDICATED JULY 28, 2005

The Anatomy of a Large-Scale Hypertextual Web Search Engine

The Anatomy of a Large-Scale Hypertextual Web Search Engine

Brin, S. and Page, L. (1998) *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* In: Seventh International World-Wide Web Conference (WWW 1998), April 14-18, 1998, Brisbane, Australia.

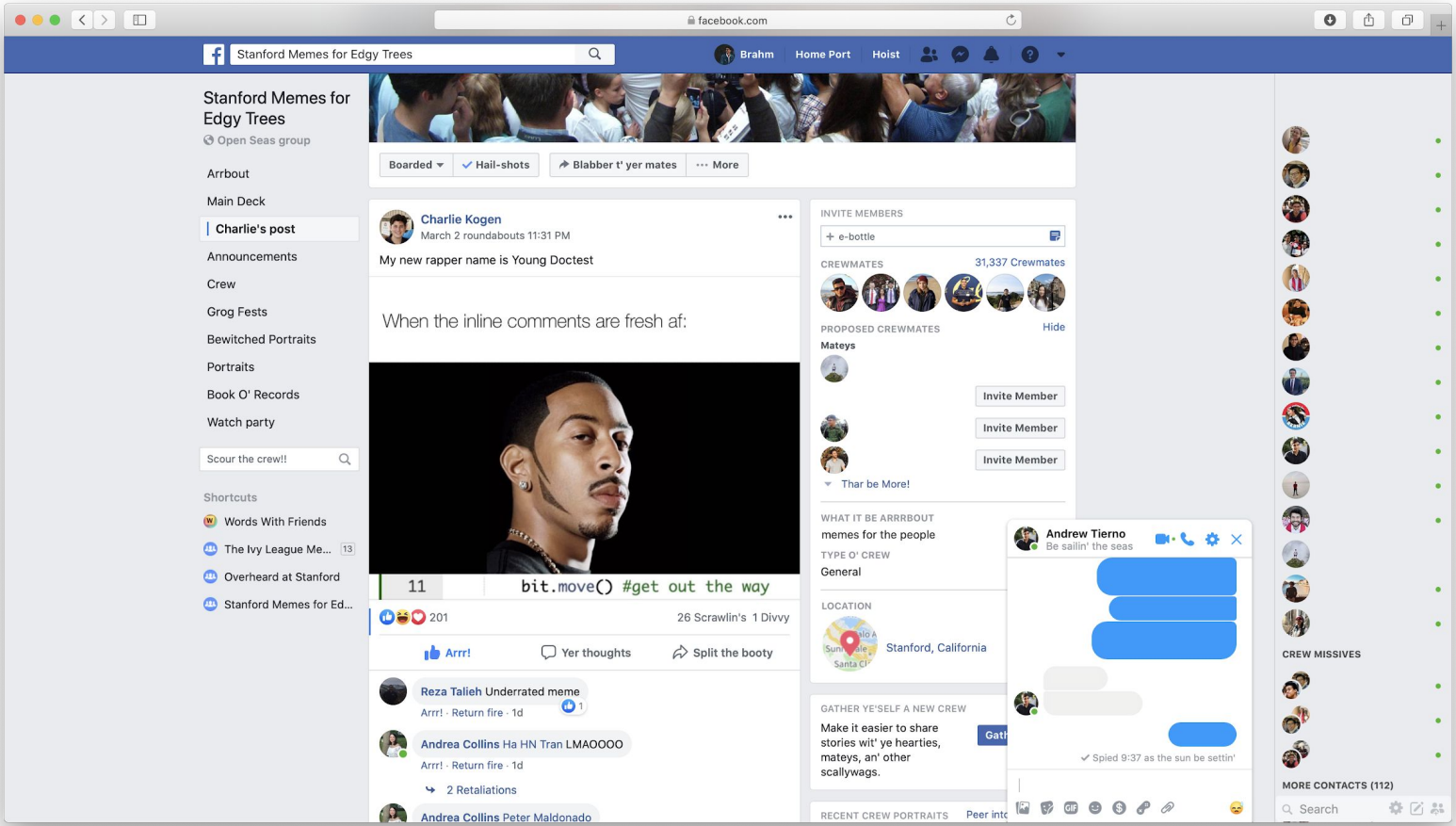
The Anatomy of a Large-Scale Hypertextual Web Search Engine

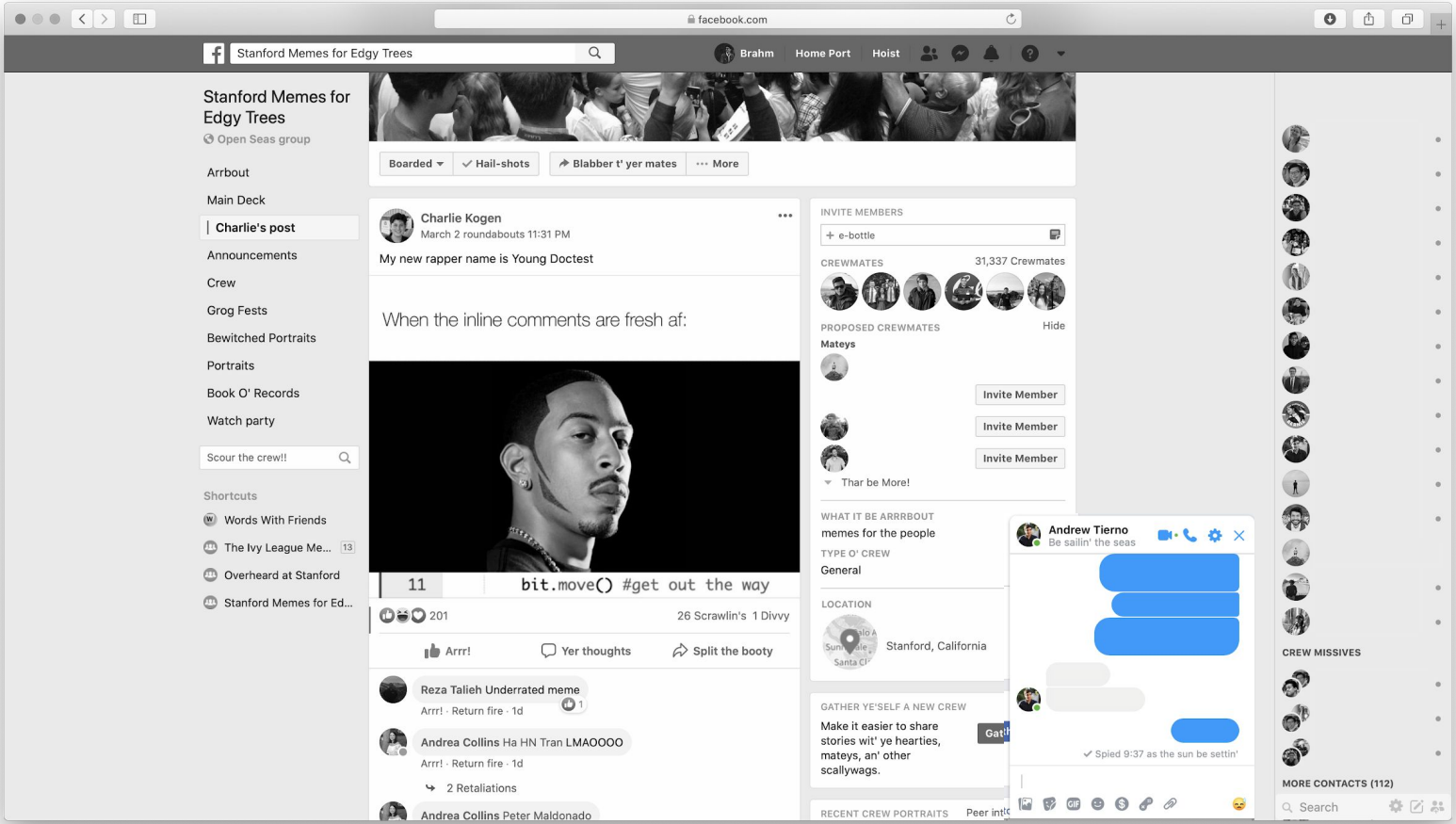
Brin, S. and Page, L. (1998) *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* In: Seventh International World-Wide Web Conference (WWW 1998), April 14-18, 1998, Brisbane, Australia.

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/. To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale web search engine -- the first such detailed public description we know of to date. Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want.


Google Storage Server
Professor Joshua Lederberg, Nobel laureate in Genetics at Stanford, having used the ARPAnet, foresaw the promise of computer-mediated communication in IEEE 1978: Digital Communications and the Conduct of Science: the New Literacy.
It took time for technology to match Lederberg's vision.
Between 1981 and 1990 NSF supported research networks, combining the ARPAnet and other services into the Internet. In 1991 Maria Zemankova, Ed Fox, and Michael Lesk issued a report on Digital Libraries. An Interagency Digital Library Initiative started in 1993 with funding by:
(D)ARPA–Brian Boesch and Gio Wiederhold (on leave from Stanford 1992-1995),
NSF–Larry Rosenberg, later Su-Shing Chen, Steve Griffin,
NASA–Paul Hunter.
Six universities received research grants, including Stanford.
In 1996, PhD students Sergey Brin and Larry Page invented the PageRank algorithm. Their work in this area was supported by Professors Hector Garcia-Molina, Rajeev Motwani, Jeffrey Ullman, and Terry Winograd. The algorithm ranks Web pages that match search terms by assessing breadth and depth of public interest in that Web content. Because much disk space was needed to test PageRank on actual World-Wide-Web data, Sergey and Larry assembled 10 of the largest drives available (4 gigabytes each) into this low-cost cabinet, whimsically decorated with Lego bricks. Their software could then index 24 million pages a week. (At the time, a massive number).
As a result of their initial work at Stanford, the two founded a company called Google in 1998. By 2008, Google reported that its thousands of servers had indexed more than a trillion unique pages.
Digital Libraries are incorporated in most public and professional services world-wide. Digital content and access has replaced much of Stanford's paper research libraries usage.


facebook.com
Stanford Memes for Edgy Trees
Brahm
Home Port
Hoist
Stanford Memes for Edgy Trees
Open Seas group
Arrbout
Main Deck
Charlie's post
Announcements
Crew
Grog Fests
Bewitched Portraits
Portraits
Book O' Records
Watch party
Scour the crew!!
Shortcuts
Words With Friends
The Ivy League Me...
Overheard at Stanford
Stanford Memes for Ed...
Boarded
Hail-shots
Blabber t' yer mates
More
Charlie Kogen
March 2 roundabouts 11:31 PM
My new rapper name is Young Doctest
When the inline comments are fresh af:
11
bit.move() #get out the way
201
26 Scrawlin's 1 Divvy
Arrr!
Yer thoughts
Split the booty
Reza Talieh Underrated meme
Arrr! · Return fire · 1d
Andrea Collins Ha HN Tran LMAOOOO
Arrr! · Return fire · 1d
2 Retaliations
Andrea Collins Peter Maldonado
INVITE MEMBERS
e-bottle
CREWMATES
31,337 Crewmates
PROPOSED CREWMATES
Hide
Mateys
Invite Member
Invite Member
Invite Member
Thar be More!
WHAT IT BE ARRRBOUT
memes for the people
TYPE O' CREW
General
LOCATION
Stanford, California
GATHER YE'SELF A NEW CREW
Make it easier to share stories wit' ye hearties, mateys, an' other scallywags.
RECENT CREW PORTRAITS
Andrew Tierno
Be sailin' the seas
Spied 9:37 as the sun be settin'
CREW MISSIVES
MORE CONTACTS (112)
Search


facebook.com
Stanford Memes for Edgy Trees
Brahm
Home Port
Hoist
Stanford Memes for Edgy Trees
Open Seas group
Arrbout
Main Deck
Charlie's post
Announcements
Crew
Grog Fests
Bewitched Portraits
Portraits
Book O' Records
Watch party
Scour the crew!!
Shortcuts
Words With Friends
The Ivy League Me...
Overheard at Stanford
Stanford Memes for Ed...
Boarded
Hail-shots
Blabber t' yer mates
More
Charlie Kogen
March 2 roundabouts 11:31 PM
My new rapper name is Young Doctest
When the inline comments are fresh af:
11
bit.move() #get out the way
201
26 Scrawlin's 1 Divvy
Arrr!
Yer thoughts
Split the booty
Reza Talieh Underrated meme
Arrr! · Return fire · 1d
Andrea Collins Ha HN Tran LMAOOOO
Arrr! · Return fire · 1d
2 Retaliations
Andrea Collins Peter Maldonado
INVITE MEMBERS
e-bottle
CREWMATES
31,337 Crewmates
PROPOSED CREWMATES
Hide
Mateys
Invite Member
Invite Member
Invite Member
Thar be More!
WHAT IT BE ARRRBOUT
memes for the people
TYPE O' CREW
General
LOCATION
Stanford, California
GATHER YE'SELF A NEW CREW
Make it easier to share stories wit' ye hearties, mateys, an' other scallywags.
RECENT CREW PORTRAITS
Andrew Tierno
Be sailin' the seas
Spied 9:37 as the sun be settin'
CREW MISSIVES
MORE CONTACTS (112)
Search

How does a chat application work?

Let's pretend Nick and I want to chat

How does a chat application work?

Both Nick and I have a chat app on our phones, which are generically referred to as chat clients

How does a chat application work?

We'll be communicating via a third computer, called a server

How does a chat application work?

Aside: in Wednesday's lecture, we saw another example of servers and clients in action.

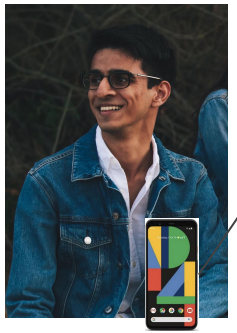
Our web browsers—and our Python programs—both acted as clients, requesting the HTML of various webpages from servers on the other end of the internet. Your browser does a ton of work to make the website look fancy, and your Python program just displays the raw HTML.

In general, a client is any program that requests information over the internet, and a server is any program that furnishes clients with that information.

How does a chat application work?

Whenever I send a message to Nick, I'm actually sending it to the server

Let's give everyone an A!



How does a chat application work?

```
Nick: hi!
Brahm: hey!
...
Brahm: Let's give
everyone an A!
```

The server appends this message to a log containing *all* our messages

How does a chat application work?

Now, when Nick next opens up his chat app, he requests our entire message history from the server

I want my messages with Brahm

How does a chat application work?

```
Nick: hi!
Brahm: hey!
...
Brahm: Let's give
everyone an A!
```

The server retrieves our message log...

How does a chat application work?

Nick: hi!
Brahm: hey!
...
Brahm: Let's give everyone an A!

... and sends it down to Nick...

How does a chat application work?

...and now Nick has the message I sent him...

Let's give everyone an A!

How does a chat application work?

...and he can respond in exactly the same way!

Let’s do it!

How does a chat application work?

```
Nick: hi!
Brahm: hey!
...
Brahm: Let's give
everyone an A!
Nick: Let's do
it!
```

This is a *very* simplified model of chat applications (come chat afterwards if you'd like to learn about how things like encryption and modern-day apps work!), but shows us the three important parts of a chat application:

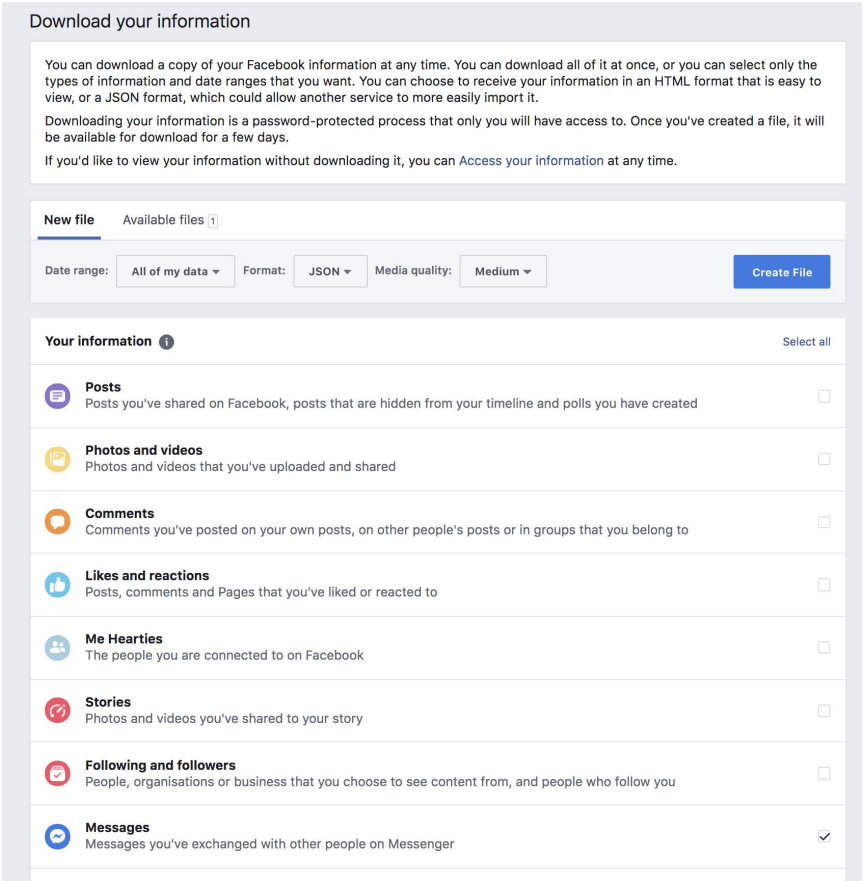
1. The client
2. The server
3. The log

How does a chat application work?

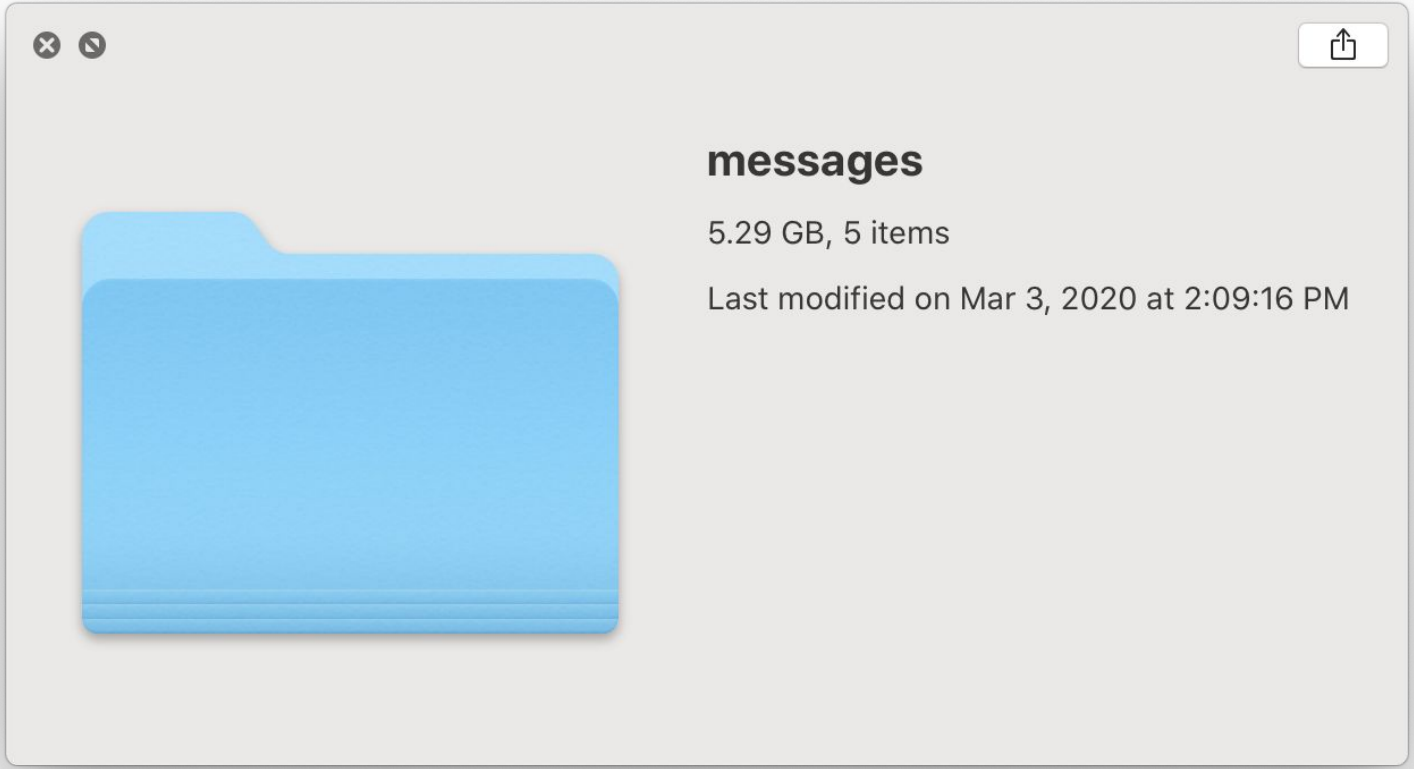
```
Nick: hi!
Brahm: hey!
...
Brahm: Let’s give everyone an A!
Nick: Let’s do it!
```

There are many fascinating things about this system, but to me, the most interesting is the fact that the log of all our messages can be stored in a file, and therefore analyzed.

It would be great if I could get access to it...



facebook.com/dyi


messages
5.29 GB, 5 items
Last modified on Mar 3, 2020 at 2:09:16 PM

The first facebook message I ever received:

“ur profile pic is lame and u have one friend. lol!! [...]”

The first facebook message I ever received:

“ur profile pic is lame and u have one friend. [...]”

Since then, I have sent or received a little less than 400,000 messages.

The first facebook message I ever received:

“ur profile pic is lame and u have one friend. [...]”

Since then, I have sent or received a little less than 400,000 messages.

Let’s have some fun.

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.

Extra slide

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 :)

Extra slide

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` (the file you just downloaded)
- 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. Set `ANONYMIZE` to `False` to deanonymize your friends.
- Open the `inbox/` folder in your terminal, and run this command:

```
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.

Extra slide

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
```

Your next assignment

Zeitgeist | *zeit·geist* | /ˈtsītˌgīst,ˈzītˌgīst/

The general intellectual, moral and cultural climate of an era

Zeitgeist | *zeit·geist* | /ˈtsītˌgīst,ˈzītˌgīst/

The general intellectual, moral and cultural climate of an era

Our question to you: can we take the enormous amount of data we have available to us and measure this?

Your tools

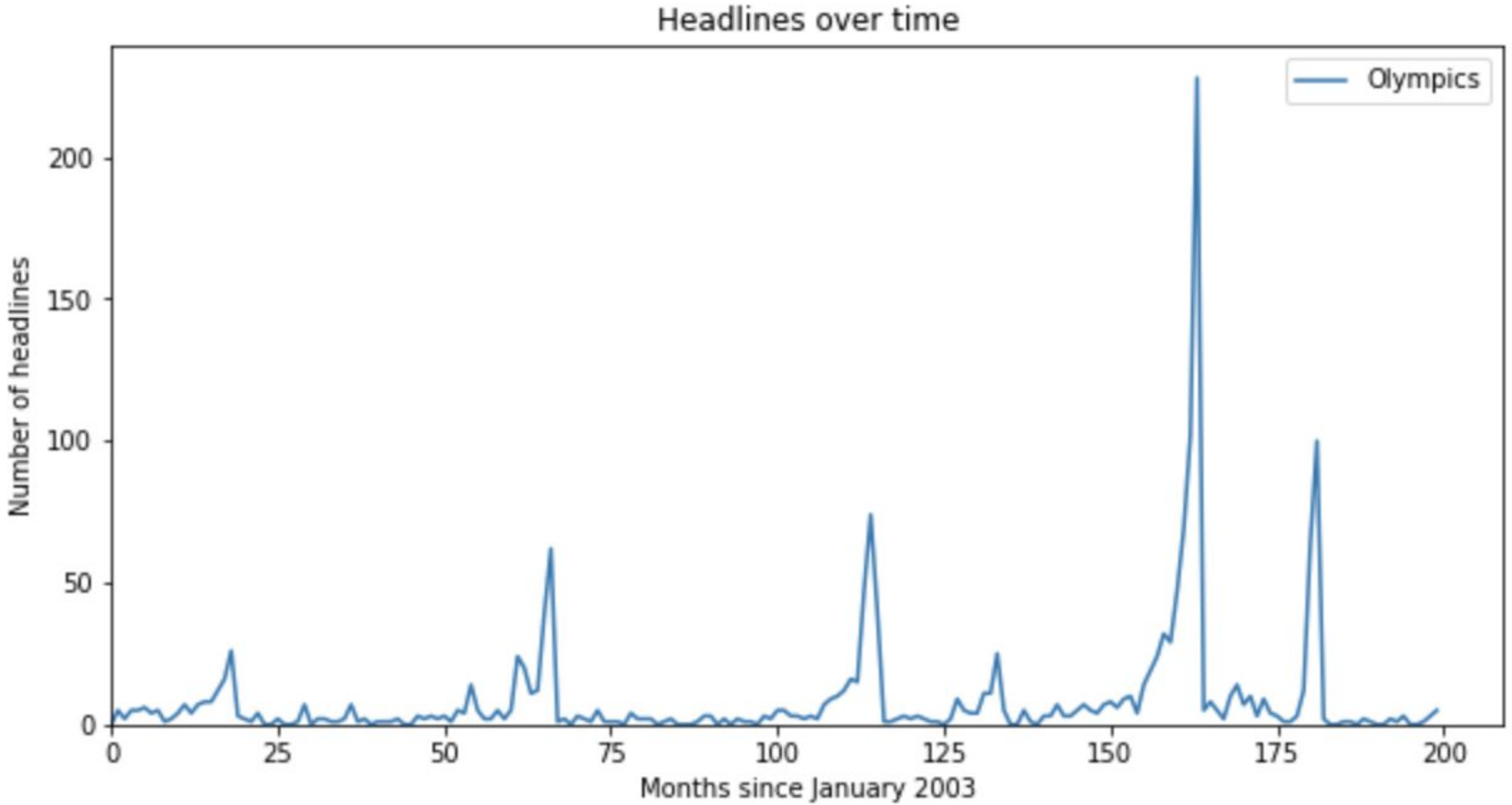
1.5 Million News Headlines

Jupyter

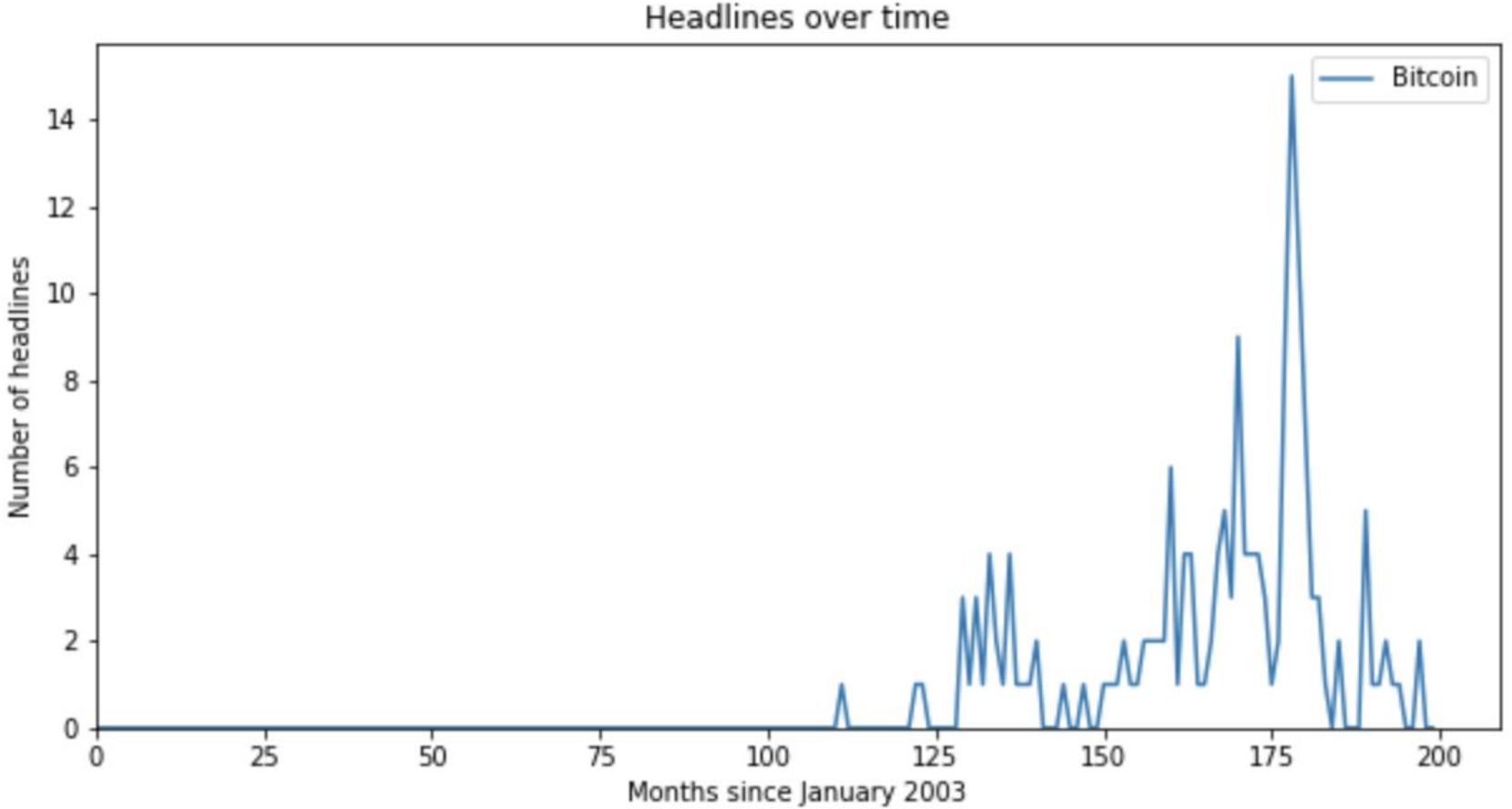
Python

Matplotlib

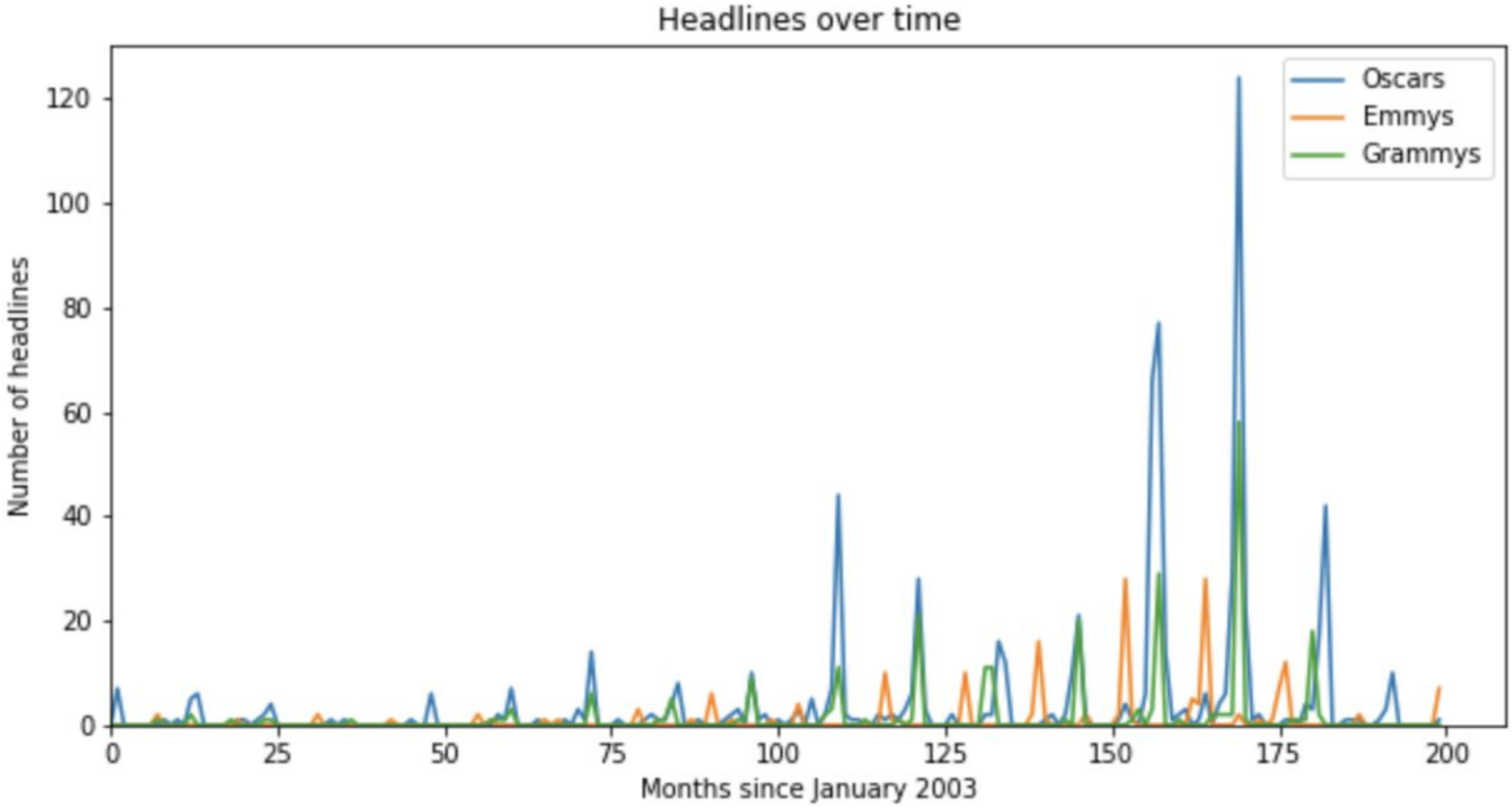
```
Type a space-separated list of search terms and press enter: Olympics
```



```
Type a space-separated list of search terms and press enter: Bitcoin
```



Type a space-separated list of search terms and press enter: Oscars Emmys Grammys



Go discover something cool!