Mining the Social Web

Mining Web Pages

This Jupyter Notebook provides an interactive way to follow along with and explore the examples from the video series. The intent behind this notebook is to reinforce the concepts in a fun, convenient, and effective way.

Using `dragnet` to extract the text from a web page

Example blog post: http://radar.oreilly.com/2010/07/louvre-industrial-age-henry-ford.html (http://radar.oreilly.com/2010/07/louvre-industrial-age-henry-ford.html)

In *Mining the Social Web, 3rd Edition*, we used a library called `boilerpipe` to extract the main content of web pages. `boilerpipe` is a sophisticated piece of software that works very well but has some software dependencies that can be very difficult to install, especially if you do not have administrative privileges on the computer you are working with. I have replaced `boilerpipe` with `Goose`, which can be easily installed using `pip`:

```
pip install goose3
```

You can learn more about `goose3` on its GitHub page (https://github.com/goose3/goose3). Another example of a content extraction library for Python is `dragnet`, which you can find here (https://github.com/dragnet-org/dragnet).

In [1]:
```
from goose3 import Goose

g = Goose()
URL='https://www.oreilly.com/ideas/ethics-in-data-project-design-its-abo
article = g.extract(url=URL)

print(article.title)
print('-'*len(article.title))
print(article.meta_description)

content = article.cleaned_text
print()
print('{}...'.format(content[:500]))
```

```
Ethics in data project design: It's about planning
--------------------------------------------------
The destination and rules of the road are clear; the route you choose to ge
t there makes a huge difference.

When I explain the value of ethics to students and professionals alike, I r
efer it as an “orientation.” As any good designer, scientist, or researcher
knows, how you orient yourself toward a problem can have a big impact on th
e sort of solution you develop—and how you get there. As Ralph Waldo Emerso
n once wrote, “perception is not whimsical, but fatal.” Your particular per
spective, knowledge of, and approach to a problem shapes your solution, ope
ning up certain paths forward and forestalling ot...
```

Using feedparser to extract the text (and other fields) from an RSS or Atom feed

In [3]:

```python
import feedparser # pip install feedparser

FEED_URL='http://feeds.feedburner.com/oreilly/radar/atom'

fp = feedparser.parse(FEED_URL)

for e in fp.entries:
    print(e.title)
    print(e.links[0].href)
    print(e.content[0].value)
```

```
Four short links: 30 April 2020
http://feedproxy.google.com/~r/oreilly/radar/atom/~3/-jpqQW9V448/ (htt
p://feedproxy.google.com/~r/oreilly/radar/atom/~3/-jpqQW9V448/)
<ol>
<li><a href="https://www.infoq.com/news/2020/04/microservices-back-agai
n/">To Microservices and Back Again: Why Segment Went Back to a Monolith
</a> &#8212; <i>microservices came with increased operational overhead an
d problems around code reuse. &#8230; If microservices are implemented in
correctly or used as a band-aid without addressing some of the root flaws
in your system, you&#8217;ll be unable to do new product development beca
use you&#8217;re drowning in the complexity.</i></li>
<li><a href="https://www.gnu.org/software/poke/">GNU poke</a> &#8212; <i>
interactive editor for binary data. Not limited to editing basic entities
such as bits and bytes, it provides a full-fledged procedural, interactiv
e programming language designed to describe data structures and to operat
e on them.</i> (via <a href="https://kernel-recipes.org/en/2019/talks/gnu
-poke-an-extensible-editor-for-structured-binary-data/">Kernel Recipes</a
>)</li>
<li><a href="https://parl.ai/projects/blender/">Blender</a> &#8212; Faceb
```

Harvesting blog data by parsing feeds

In [4]:

```
import os
import sys
import json
import feedparser
from bs4 import BeautifulSoup
from nltk import clean_html

FEED_URL = 'http://feeds.feedburner.com/oreilly/radar/atom'

def cleanHtml(html):
    if html == "": return ""

    return BeautifulSoup(html, 'html5lib').get_text()

fp = feedparser.parse(FEED_URL)

print("Fetched {0} entries from '{1}'".format(len(fp.entries[0].title),

blog_posts = []
for e in fp.entries:
    blog_posts.append({'title': e.title, 'content'
                      : cleanHtml(e.content[0].value), 'link': e.links[0

out_file = os.path.join('feed.json')
f = open(out_file, 'w+')
f.write(json.dumps(blog_posts, indent=1))
f.close()

print('Wrote output file to {0}'.format(f.name))
```

```
Fetched 31 entries from 'Radar'
Wrote output file to feed.json
```

Starting to write a web crawler

In [6]:

```python
import httplib2
import re
from bs4 import BeautifulSoup

http = httplib2.Http()
status, response = http.request('http://www.nytimes.com')

soup = BeautifulSoup(response, 'html5lib')

links = []

for link in soup.findAll('a', attrs={'href': re.compile("^http(s?)://")}
    links.append(link.get('href'))

for link in links:
    print(link)
```

```
https://www.nytimes.com/es/ (https://www.nytimes.com/es/)
https://cn.nytimes.com (https://cn.nytimes.com)
https://myaccount.nytimes.com/auth/login?response_type=cookie&client_id=vi
 (https://myaccount.nytimes.com/auth/login?response_type=cookie&client_id=v
i)
https://www.nytimes.com/section/todayspaper (https://www.nytimes.com/sectio
n/todayspaper)
https://www.nytimes.com/section/world (https://www.nytimes.com/section/worl
d)
https://www.nytimes.com/section/us (https://www.nytimes.com/section/us)
https://www.nytimes.com/section/politics (https://www.nytimes.com/section/p
olitics)
https://www.nytimes.com/section/nyregion (https://www.nytimes.com/section/n
yregion)
https://www.nytimes.com/section/business (https://www.nytimes.com/section/b
usiness)
https://www.nytimes.com/section/opinion (https://www.nytimes.com/section/op
inion)
https://www.nytimes.com/section/technology (https://www.nytimes.com/sectio
n/technology)
https://www.nytimes.com/section/science (https://www.nytimes.com/section/sc
ience)
https://www.nytimes.com/section/health (https://www.nytimes.com/section/hea
lth)
https://www.nytimes.com/section/sports (https://www.nytimes.com/section/spo
rts)
https://www.nytimes.com/section/arts (https://www.nytimes.com/section/arts)
https://www.nytimes.com/section/books (https://www.nytimes.com/section/book
s)
https://www.nytimes.com/section/style (https://www.nytimes.com/section/styl
e)
https://www.nytimes.com/section/food (https://www.nytimes.com/section/food)
https://www.nytimes.com/section/travel (https://www.nytimes.com/section/tra
vel)
https://www.nytimes.com/section/magazine (https://www.nytimes.com/section/m
agazine)
https://www.nytimes.com/section/t-magazine (https://www.nytimes.com/sectio
n/t-magazine)
https://www.nytimes.com/section/realestate (https://www.nytimes.com/sectio
n/realestate)
https://www.nytimes.com/video (https://www.nytimes.com/video)
https://www.nytimes.com/section/world (https://www.nytimes.com/section/worl
d)
https://www.nytimes.com/section/us (https://www.nytimes.com/section/us)
https://www.nytimes.com/section/politics (https://www.nytimes.com/section/p
olitics)
https://www.nytimes.com/section/nyregion (https://www.nytimes.com/section/n
yregion)
https://www.nytimes.com/section/business (https://www.nytimes.com/section/b
usiness)
https://www.nytimes.com/section/opinion (https://www.nytimes.com/section/op
inion)
https://www.nytimes.com/section/technology (https://www.nytimes.com/sectio
n/technology)
https://www.nytimes.com/section/science (https://www.nytimes.com/section/sc
ience)
https://www.nytimes.com/section/health (https://www.nytimes.com/section/hea
lth)
https://www.nytimes.com/section/sports (https://www.nytimes.com/section/spo
rts)
https://www.nytimes.com/section/arts (https://www.nytimes.com/section/arts)
https://www.nytimes.com/section/books (https://www.nytimes.com/section/book
s)
https://www.nytimes.com/section/style (https://www.nytimes.com/section/styl
e)
https://www.nytimes.com/section/food (https://www.nytimes.com/section/food)
https://www.nytimes.com/section/travel (https://www.nytimes.com/section/tra
vel)
https://www.nytimes.com/section/magazine (https://www.nytimes.com/section/m
agazine)
https://www.nytimes.com/section/t-magazine (https://www.nytimes.com/sectio
n/t-magazine)
https://www.nytimes.com/section/realestate (https://www.nytimes.com/sectio
n/realestate)
https://www.nytimes.com/video (https://www.nytimes.com/video)
https://www.nytimes.com/newsletters/watching (https://www.nytimes.com/newsl
etters/watching)
https://www.nytimes.com/newsletters/watching (https://www.nytimes.com/newsl
etters/watching)
https://www.nytimes.com/2020/05/03/us/coronavirus-updates.html?type=styln-l
ive-updates&label=u.s.&index=0 (https://www.nytimes.com/2020/05/03/us/coron
avirus-updates.html?type=styln-live-updates&label=u.s.&index=0)
https://www.nytimes.com/2020/05/03/us/coronavirus-updates.html?type=styln-l
ive-updates&label=u.s.&index=0#link-1333a84e (https://www.nytimes.com/2020/
05/03/us/coronavirus-updates.html?type=styln-live-updates&label=u.s.&index=
0#link-1333a84e)
https://www.nytimes.com/2020/05/03/us/coronavirus-updates.html?type=styln-l
ive-updates&label=u.s.&index=0#link-74d5cf12 (https://www.nytimes.com/2020/
05/03/us/coronavirus-updates.html?type=styln-live-updates&label=u.s.&index=
0#link-74d5cf12)
https://www.nytimes.com/2020/05/03/us/coronavirus-updates.html?type=styln-l
ive-updates&label=u.s.&index=0#link-e19c29d (https://www.nytimes.com/2020/0
5/03/us/coronavirus-updates.html?type=styln-live-updates&label=u.s.&index=0
#link-e19c29d)
https://www.nytimes.com/2020/05/03/world/coronavirus-news.html?type=styln-l
ive-updates&label=global&index=1 (https://www.nytimes.com/2020/05/03/world/
coronavirus-news.html?type=styln-live-updates&label=global&index=1)
https://www.nytimes.com/2020/05/03/world/coronavirus-news.html?type=styln-l
ive-updates&label=global&index=1#link-7d1e0ed1 (https://www.nytimes.com/202
0/05/03/world/coronavirus-news.html?type=styln-live-updates&label=global&in
dex=1#link-7d1e0ed1)
https://www.nytimes.com/2020/05/03/world/coronavirus-news.html?type=styln-l
ive-updates&label=global&index=1#link-7b7dbdc0 (https://www.nytimes.com/202
0/05/03/world/coronavirus-news.html?type=styln-live-updates&label=global&in
dex=1#link-7b7dbdc0)
https://www.nytimes.com/2020/05/03/world/coronavirus-news.html?type=styln-l
ive-updates&label=global&index=1#link-6a5b6857 (https://www.nytimes.com/202
0/05/03/world/coronavirus-news.html?type=styln-live-updates&label=global&in
dex=1#link-6a5b6857)
https://www.nytimes.com/2020/05/03/nyregion/coronavirus-new-york-update.htm
l?type=styln-live-updates&label=new (https://www.nytimes.com/2020/05/03/nyr
egion/coronavirus-new-york-update.html?type=styln-live-updates&label=new) y
ork&index=2
https://www.nytimes.com/2020/05/03/nyregion/coronavirus-new-york-update.htm
l?type=styln-live-updates&label=new (https://www.nytimes.com/2020/05/03/nyr
egion/coronavirus-new-york-update.html?type=styln-live-updates&label=new) y
ork&index=2#link-792b9709
https://www.nytimes.com/2020/05/03/nyregion/coronavirus-new-york-update.htm
l?type=styln-live-updates&label=new (https://www.nytimes.com/2020/05/03/nyr
egion/coronavirus-new-york-update.html?type=styln-live-updates&label=new) y
ork&index=2#link-3a1298a0
https://www.nytimes.com/2020/05/03/nyregion/coronavirus-new-york-update.htm
l?type=styln-live-updates&label=new (https://www.nytimes.com/2020/05/03/nyr
egion/coronavirus-new-york-update.html?type=styln-live-updates&label=new) y
ork&index=2#link-5abbec4a
https://www.nytimes.com/news-event/coronavirus (https://www.nytimes.com/new
s-event/coronavirus)
https://www.nytimes.com/2020/05/01/health/coronavirus-covid-toe.html (http
s://www.nytimes.com/2020/05/01/health/coronavirus-covid-toe.html)
https://www.nytimes.com/2020/04/29/well/coronavirus-exercise-heart-health.h
tml (https://www.nytimes.com/2020/04/29/well/coronavirus-exercise-heart-hea
lth.html)
https://www.nytimes.com/2020/04/30/well/live/coronavirus-days-5-through-10.
html (https://www.nytimes.com/2020/04/30/well/live/coronavirus-days-5-throu
gh-10.html)
https://www.nytimes.com/section/opinion?pagetype=Homepage&action=click&modu
le=Opinion (https://www.nytimes.com/section/opinion?pagetype=Homepage&actio
n=click&module=Opinion)
http://nyt.qualtrics.com/jfe/form/SV_eFJmKj9v0krSE0l (http://nyt.qualtrics.
com/jfe/form/SV_eFJmKj9v0krSE0l)
https://help.nytimes.com/hc/en-us/articles/115014792127-Copyright-notice (h
ttps://help.nytimes.com/hc/en-us/articles/115014792127-Copyright-notice)
https://www.nytco.com/ (https://www.nytco.com/)
https://help.nytimes.com/hc/en-us/articles/115015385887-Contact-Us (http
s://help.nytimes.com/hc/en-us/articles/115015385887-Contact-Us)
https://www.nytco.com/careers/ (https://www.nytco.com/careers/)
https://nytmediakit.com/ (https://nytmediakit.com/)
http://www.tbrandstudio.com/ (http://www.tbrandstudio.com/)
https://help.nytimes.com/hc/en-us/articles/115014892108-Privacy-policy (htt
ps://help.nytimes.com/hc/en-us/articles/115014892108-Privacy-policy)
https://help.nytimes.com/hc/en-us/articles/115014892108-Privacy-policy (htt
ps://help.nytimes.com/hc/en-us/articles/115014892108-Privacy-policy)
https://help.nytimes.com/hc/en-us/articles/115014893428-Terms-of-service (h
ttps://help.nytimes.com/hc/en-us/articles/115014893428-Terms-of-service)
https://help.nytimes.com/hc/en-us/articles/115014893968-Terms-of-sale (http
s://help.nytimes.com/hc/en-us/articles/115014893968-Terms-of-sale)
https://spiderbites.nytimes.com (https://spiderbites.nytimes.com)
https://help.nytimes.com/hc/en-us (https://help.nytimes.com/hc/en-us)
https://www.nytimes.com/subscription?campaignId=37WXW (https://www.nytimes.
com/subscription?campaignId=37WXW)
```

```
Create an empty graph
Create an empty queue to keep track of nodes that need to be processed

Add the starting point to the graph as the root node
Add the root node to a queue for processing

Repeat until some maximum depth is reached or the queue is empty:
  Remove a node from the queue
  For each of the node's neighbors:
    If the neighbor hasn't already been processed:
      Add it to the queue
      Add it to the graph
      Create an edge in the graph that connects the node and its neighbo
r
```

Using NLTK to parse web page data

Naive sentence detection based on periods

In [7]:

```
text = "Mr. Green killed Colonel Mustard in the study with the candlesti
print(text.split("."))
```

```
['Mr', ' Green killed Colonel Mustard in the study with the candlestick', '
Mr', ' Green is not a very nice fellow', '']
```

More sophisticated sentence detection

In [8]:

```
import nltk # Installation instructions: http://www.nltk.org/install.htm

# Downloading nltk packages used in this example
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\szaki5\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[8]: True

In [9]:

```
sentences = nltk.tokenize.sent_tokenize(text)
print(sentences)
```

```
['Mr. Green killed Colonel Mustard in the study with the candlestick.', 'M
r. Green is not a very nice fellow.']
```

In [10]:

```
harder_example = """My name is John Smith and my email address is j.smit
Mostly people call Mr. Smith. But I actually have a Ph.D.!
Can you believe it? Neither can most people..."""

sentences = nltk.tokenize.sent_tokenize(harder_example)
print(sentences)
```

```
['My name is John Smith and my email address is j.smith@company.com.', 'Mos
tly people call Mr. Smith.', 'But I actually have a Ph.D.!', 'Can you belie
ve it?', 'Neither can most people...']
```

Word tokenization

In [11]:

```
text = "Mr. Green killed Colonel Mustard in the study with the candlesti
sentences = nltk.tokenize.sent_tokenize(text)

tokens = [nltk.word_tokenize(s) for s in sentences]
print(tokens)
```

```
[['Mr.', 'Green', 'killed', 'Colonel', 'Mustard', 'in', 'the', 'study', 'wi
th', 'the', 'candlestick', '.'], ['Mr.', 'Green', 'is', 'not', 'a', 'very',
'nice', 'fellow', '.']]
```

Part of speech tagging for tokens

In [12]:

```
import nltk
nltk.download('averaged_perceptron_tagger')

# Downloading nltk packages used in this example
nltk.download('maxent_treebank_pos_tagger')

pos_tagged_tokens = [nltk.pos_tag(t) for t in tokens]
print(pos_tagged_tokens)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\szaki5\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
[nltk_data] Downloading package maxent_treebank_pos_tagger to
[nltk_data]     C:\Users\szaki5\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\maxent_treebank_pos_tagger.zip.
```

```
[[('Mr.', 'NNP'), ('Green', 'NNP'), ('killed', 'VBD'), ('Colonel', 'NNP'),
('Mustard', 'NNP'), ('in', 'IN'), ('the', 'DT'), ('study', 'NN'), ('with',
'IN'), ('the', 'DT'), ('candlestick', 'NN'), ('.', '.')], [('Mr.', 'NNP'),
('Green', 'NNP'), ('is', 'VBZ'), ('not', 'RB'), ('a', 'DT'), ('very', 'R
B'), ('nice', 'JJ'), ('fellow', 'NN'), ('.', '.')]]
```

Alphabetical list of part-of-speech tags used in the Penn Treebank Project

See: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html (https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html)

#	POS Tag	Meaning
1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential there
5	FW	Foreign word
6	IN	Preposition or subordinating conjunction
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker
11	MD	Modal
12	NN	Noun, singular or mass
13	NNS	Noun, plural
14	NNP	Proper noun, singular
15	NNPS	Proper noun, plural
16	PDT	Predeterminer
17	POS	Possessive ending
18	PRP	Personal pronoun
19	PRP$	Possessive pronoun
20	RB	Adverb
21	RBR	Adverb, comparative
22	RBS	Adverb, superlative
23	RP	Particle
24	SYM	Symbol
25	TO	to
26	UH	Interjection
27	VB	Verb, base form
28	VBD	Verb, past tense
29	VBG	Verb, gerund or present participle
30	VBN	Verb, past participle
31	VBP	Verb, non-3rd person singular present
32	VBZ	Verb, 3rd person singular present
33	WDT	Wh-determiner

#	POS Tag	Meaning
34	WP	Wh-pronoun
35	WP$	Possessive wh-pronoun
36	WRB	Wh-adverb

Named entity extraction/chunking for tokens

In [13]:
```
# Downloading nltk packages used in this example
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\szaki5\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping chunkers\maxent_ne_chunker.zip.
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\szaki5\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\words.zip.
```

Out[13]:
```
True
```

In [14]:
```
jim = "Jim bought 300 shares of Acme Corp. in 2006."

tokens = nltk.word_tokenize(jim)
jim_tagged_tokens = nltk.pos_tag(tokens)

ne_chunks = nltk.chunk.ne_chunk(jim_tagged_tokens)
```

In [15]:
```
ne_chunks
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
C:\Anaconda\lib\site-packages\IPython\core\formatters.py in __call__(self,
 obj)
    343             method = get_real_method(obj, self.print_method)
    344             if method is not None:
--> 345                 return method()
    346             return None
    347         else:

C:\Anaconda\lib\site-packages\nltk\tree.py in _repr_png_(self)
    819                 raise LookupError
    820 
--> 821             with open(out_path, 'rb') as sr:
    822                 res = sr.read()
    823             os.remove(in_path)

FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\szaki5
\\AppData\\Local\\Temp\\tmpi9_8ro11.png'
```

Out[15]:
```
Tree('S', [Tree('PERSON', [('Jim', 'NNP')]), ('bought', 'VBD'), ('300', 'C
D'), ('shares', 'NNS'), ('of', 'IN'), Tree('ORGANIZATION', [('Acme', 'NN
P'), ('Corp.', 'NNP')]), ('in', 'IN'), ('2006', 'CD'), ('.', '.')])
```

In [16]:
```
ne_chunks = [nltk.chunk.ne_chunk(ptt) for ptt in pos_tagged_tokens]

ne_chunks[0].pprint()
ne_chunks[1].pprint()
```

```
(S
  (PERSON Mr./NNP)
  (PERSON Green/NNP)
  killed/VBD
  (ORGANIZATION Colonel/NNP Mustard/NNP)
  in/IN
  the/DT
  study/NN
  with/IN
  the/DT
  candlestick/NN
  ./.)
(S
  (PERSON Mr./NNP)
  (ORGANIZATION Green/NNP)
  is/VBZ
  not/RB
  a/DT
  very/RB
  nice/JJ
  fellow/NN
  ./.)
```

Using NLTK’s NLP tools to process human language in blog data

In [18]:

```python
import json
import nltk

BLOG_DATA = "data/feed.json"

blog_data = json.loads(open(BLOG_DATA).read())

# Download nltk packages used in this example
nltk.download('stopwords')

# Customize your list of stopwords as needed. Here, we add common
# punctuation and contraction artifacts.

stop_words = nltk.corpus.stopwords.words('english') + [
    '.',
    ',',
    '--',
    '\'s',
    '?',
    ')',
    '(',
    ':',
    '\'',
    '\'re',
    '"',
    '-',
    '}',
    '{',
    u'—',
    ']',
    '[',
    '...'
    ]
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\szaki5\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [19]:

```python
for post in blog_data:
    sentences = nltk.tokenize.sent_tokenize(post['content'])

    words = [w.lower() for sentence in sentences for w in
             nltk.tokenize.word_tokenize(sentence)]

    fdist = nltk.FreqDist(words)

    # Remove stopwords from fdist
    for sw in stop_words:
        del fdist[sw]

    # Basic stats

    num_words = sum([i[1] for i in fdist.items()])
    num_unique_words = len(fdist.keys())

    # Hapaxes are words that appear only once
    num_hapaxes = len(fdist.hapaxes())

    top_10_words_sans_stop_words = fdist.most_common(10)

    print(post['title'])
    print('\tNum Sentences:'.ljust(25), len(sentences))
    print('\tNum Words:'.ljust(25), num_words)
    print('\tNum Unique Words:'.ljust(25), num_unique_words)
    print('\tNum Hapaxes:'.ljust(25), num_hapaxes)
    print('\tTop 10 Most Frequent Words (sans stop words):\n\t\t', \
          '\n\t\t'.join(['{0} ({1})'.format(w[0], w[1]) for w in top_10_
    print()
```

```
                 ’ (21)
                “ (21)

                ” (21)
                conversational (15)
                bots (7)
                says (7)
                interaction (7)
                must (7)
                user (7)
                kai (7)

Four short links: 18 August 2017
        Num Sentences:           16
        Num Words:               263
        Num Unique Words:        204
        Num Hapaxes:             173
        Top 10 Most Frequent Words (sans stop words):
                 hype (9)
                jobs (5)
```

A document summarization algorithm based principally upon sentence detection and frequency analysis within sentences

In [20]:

```
import json
import nltk
import numpy

BLOG_DATA = "feed.json"

blog_data = json.loads(open(BLOG_DATA).read())

N = 100  # Number of words to consider
CLUSTER_THRESHOLD = 5  # Distance between words to consider
TOP_SENTENCES = 5  # Number of sentences to return for a "top n" summary
```

In [21]:

```
stop_words = nltk.corpus.stopwords.words('english') + [
    '.',
    ',',
    '--',
    '\'s',
    '?',
    ')',
    '(',
    ':',
    '\'',
    '\'re',
    '"',
    '-',
    '}',
    '{',
    u'—',
    '>',
    '<',
    '...'
    ]
```

In [22]:

```
# Approach taken from "The Automatic Creation of Literature Abstracts" b
def _score_sentences(sentences, important_words):
    scores = []
    sentence_idx = 0

    for s in [nltk.tokenize.word_tokenize(s) for s in sentences]:

        word_idx = []

        # For each word in the word list...
        for w in important_words:
            try:
                # Compute an index for where any important words occur i
                word_idx.append(s.index(w))
            except ValueError: # w not in this particular sentence
                pass

        word_idx.sort()

        # It is possible that some sentences may not contain any importa
        if len(word_idx)== 0: continue

        # Using the word index, compute clusters by using a max distance
        # for any two consecutive words.

        clusters = []
        cluster = [word_idx[0]]
        i = 1
        while i < len(word_idx):
            if word_idx[i] - word_idx[i - 1] < CLUSTER_THRESHOLD:
                cluster.append(word_idx[i])
            else:
                clusters.append(cluster[:])
                cluster = [word_idx[i]]
            i += 1
        clusters.append(cluster)

        # Score each cluster. The max score for any given cluster is the
        # for the sentence.

        max_cluster_score = 0

        for c in clusters:
            significant_words_in_cluster = len(c)
            # true clusters also contain insignificant words, so we get
            # the total cluster length by checking the indices
            total_words_in_cluster = c[-1] - c[0] + 1
            score = 1.0 * significant_words_in_cluster**2 / total_words_

            if score > max_cluster_score:
                max_cluster_score = score

        scores.append((sentence_idx, max_cluster_score))
        sentence_idx += 1

    return scores
```

In [24]:

```python
def summarize(txt):
    sentences = [s for s in nltk.tokenize.sent_tokenize(txt)]
    normalized_sentences = [s.lower() for s in sentences]

    words = [w.lower() for sentence in normalized_sentences for w in
             nltk.tokenize.word_tokenize(sentence)]

    fdist = nltk.FreqDist(words)

    # Remove stopwords from fdist
    for sw in stop_words:
        del fdist[sw]

    top_n_words = [w[0] for w in fdist.most_common(N)]

    scored_sentences = _score_sentences(normalized_sentences, top_n_word

    # Summarization Approach 1:
    # Filter out nonsignificant sentences by using the average score plu
    # fraction of the std dev as a filter

    avg = numpy.mean([s[1] for s in scored_sentences])
    std = numpy.std([s[1] for s in scored_sentences])
    mean_scored = [(sent_idx, score) for (sent_idx, score) in scored_sen
                   if score > avg + 0.5 * std]

    # Summarization Approach 2:
    # Another approach would be to return only the top N ranked sentence

    top_n_scored = sorted(scored_sentences, key=lambda s: s[1])[-TOP_SEN
    top_n_scored = sorted(top_n_scored, key=lambda s: s[0])

    # Decorate the post object with summaries

    return dict(top_n_summary=[sentences[idx] for (idx, score) in top_n_
                mean_scored_summary=[sentences[idx] for (idx, score) in
```

In [25]:
```
for post in blog_data:
    post.update(summarize(post['content']))

    print(post['title'])
    print('=' * len(post['title']))
    print()
    print('Top N Summary')
    print('-------------')
    print(' '.join(post['top_n_summary']))
    print()
    print('Mean Scored Summary')
    print('-------------------')
    print(' '.join(post['mean_scored_summary']))
    print()
```

```
extbook and into the real world. Experts in a variety of fields will casu
ally converse with the hosts about how the particular game discussed appl
ies to their work. The podcast is hosted by game theorist Ben Klemens and

science journalist and composer Liz Landau. (via Ben Klemens)
Verification Handbook (3ed) — latest guide to investigating disinformatio
n and media manipulation, covering identifying actors, investigating plat
forms, tracking ads, etc. (via Craig Silverman)
Ransomware Groups (Microsoft) — analysis of ransomware campaigns yields t
his report, which includes a great graphic taxonomy of ransomware payload
s.

Mean Scored Summary
-------------------

podpaperscissors — From the classic “prisoner’s dilemma” to more obscure
coördination games, Pod Paper Scissors takes game theory out of the dry t
extbook and into the real world. (via Ben Klemens)
Verification Handbook (3ed) — latest guide to investigating disinformatio
n and media manipulation, covering identifying actors, investigating plat
forms, tracking ads, etc.
```

Visualizing document summarization results with HTML output

In [26]:

```python
import os
from IPython.display import IFrame
from IPython.core.display import display

HTML_TEMPLATE = """<html>
    <head>
        <title>{0}</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
    </head>
    <body>{1}</body>
</html>"""

for post in blog_data:

    # Uses previously defined summarize function.
    post.update(summarize(post['content']))

    # You could also store a version of the full post with key sentences
    # for analysis with simple string replacement...

    for summary_type in ['top_n_summary', 'mean_scored_summary']:
        post[summary_type + '_marked_up'] = '<p>{0}</p>'.format(post['co

        for s in post[summary_type]:
            post[summary_type + '_marked_up'] = \
            post[summary_type + '_marked_up'].replace(s, '<strong>{0}</s

        filename = post['title'].replace("?", "") + '.summary.' + summar

        f = open(os.path.join(filename), 'wb')
        html = HTML_TEMPLATE.format(post['title'] + ' Summary', post[sum
        f.write(html.encode('utf-8'))
        f.close()

        print("Data written to", f.name)

# Display any of these files with an inline frame. This displays the
# last file processed by using the last value of f.name...
print()
print("Displaying {0}:".format(f.name))
display(IFrame('files/{0}'.format(f.name), '100%', '600px'))
```

```
Data written to Four short links: 30 April 2020.summary.top_n_summary.htm
l
Data written to Four short links: 30 April 2020.summary.mean_scored_summa
ry.html
Data written to Four short links: 29 April 2020.summary.top_n_summary.htm
l
Data written to Four short links: 29 April 2020.summary.mean_scored_summa
ry.html
Data written to Four short links: 28 April 2020.summary.top_n_summary.htm
l
Data written to Four short links: 28 April 2020.summary.mean_scored_summa
ry.html
Data written to Four short links: 27 April 2020.summary.top_n_summary.htm
```

```
l
Data written to Four short links: 27 April 2020.summary.mean_scored_summa
ry.html
Data written to Four short links: 24 April 2020.summary.top_n_summary.htm
l
```

Extracting entities from a text with NLTK

In [27]:

```
import nltk
import json

BLOG_DATA = "feed.json"

blog_data = json.loads(open(BLOG_DATA).read())

for post in blog_data:

    sentences = nltk.tokenize.sent_tokenize(post['content'])
    tokens = [nltk.tokenize.word_tokenize(s) for s in sentences]
    pos_tagged_tokens = [nltk.pos_tag(t) for t in tokens]

    # Flatten the list since we're not using sentence structure
    # and sentences are guaranteed to be separated by a special
    # POS tuple such as ('.', '.')

    pos_tagged_tokens = [token for sent in pos_tagged_tokens for token i

    all_entity_chunks = []
    previous_pos = None
    current_entity_chunk = []
    for (token, pos) in pos_tagged_tokens:

        if pos == previous_pos and pos.startswith('NN'):
            current_entity_chunk.append(token)
        elif pos.startswith('NN'):

            if current_entity_chunk != []:

                # Note that current_entity_chunk could be a duplicate wh
                # so frequency analysis again becomes a consideration

                all_entity_chunks.append((' '.join(current_entity_chunk)
            current_entity_chunk = [token]

        previous_pos = pos

    # Store the chunks as an index for the document
    # and account for frequency while we're at it...

    post['entities'] = {}
    for c in all_entity_chunks:
        post['entities'][c] = post['entities'].get(c, 0) + 1

    # For example, we could display just the title-cased entities

    print(post['title'])
    print('-' * len(post['title']))
    proper_nouns = []
    for (entity, pos) in post['entities']:
        if entity.istitle():
            print('\t{0} ({1})'.format(entity, post['entities'][(entity,
    print()
```

```
Four short links: 30 April 2020
-------------------------------
        Microservices (1)
        Back Again (1)
        Segment Went Back (1)
        Monolith (1)
        Kernel Recipes (1)
        Blender – Facebook (1)
        Videos (1)

Four short links: 29 April 2020
-------------------------------
        Pod Paper Scissors (1)
        Experts (1)
        Ben Klemens (1)
        Liz Landau (1)
        Ben Klemens (1)
        Verification Handbook (1)
```

Discovering interactions between entities

In [28]:

```python
import nltk
import json

BLOG_DATA = "feed.json"

def extract_interactions(txt):
    sentences = nltk.tokenize.sent_tokenize(txt)
    tokens = [nltk.tokenize.word_tokenize(s) for s in sentences]
    pos_tagged_tokens = [nltk.pos_tag(t) for t in tokens]

    entity_interactions = []
    for sentence in pos_tagged_tokens:

        all_entity_chunks = []
        previous_pos = None
        current_entity_chunk = []

        for (token, pos) in sentence:

            if pos == previous_pos and pos.startswith('NN'):
                current_entity_chunk.append(token)
            elif pos.startswith('NN'):
                if current_entity_chunk != []:
                    all_entity_chunks.append((' '.join(current_entity_ch
                            pos))
                current_entity_chunk = [token]

            previous_pos = pos

        if len(all_entity_chunks) > 1:
            entity_interactions.append(all_entity_chunks)
        else:
            entity_interactions.append([])

    assert len(entity_interactions) == len(sentences)

    return dict(entity_interactions=entity_interactions,
                sentences=sentences)

blog_data = json.loads(open(BLOG_DATA).read())

# Display selected interactions on a per-sentence basis

for post in blog_data:

    post.update(extract_interactions(post['content']))

    print(post['title'])
    print('-' * len(post['title']))
    for interactions in post['entity_interactions']:
        print('; '.join([i[0] for i in interactions]))
    print()
```

```
football video; games; physics; football simulation; agents; football; pl
ayers; team; manage; opponent; s defense; order
```

```
Four short links: 27 April 2020
-------------------------------
Process; Different Computer; proof; concept; t replicate; things; s; fun
tech demo; ’; t use; anything; telefork; function call; process; machine;
instance
Consistency Maps – Jepsen; safety; properties; systems–most; violations;
consistency


kind; consistency
reference guide; definitions; explanations; underpinnings; consistency; m
odels; engineers
wasmachine; –; wasmachine; implementation; WebAssembly; specification

Expert Twitter Only Goes; Bring Back Blogs; Wired; re; opinion; machines;
opinion; inflammatory; fit; businesses; people
Syllabus; systems; substack; rant; response; dearth
```

Visualizing interactions between entities with HTML output

In [29]:

```
import os
import json
import nltk
from IPython.display import IFrame
from IPython.core.display import display

BLOG_DATA = "feed.json"

HTML_TEMPLATE = """<html>
    <head>
        <title>{0}</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
    </head>
    <body>{1}</body>
</html>"""

blog_data = json.loads(open(BLOG_DATA).read())

for post in blog_data:

    post.update(extract_interactions(post['content']))

    # Display output as markup with entities presented in bold text

    post['markup'] = []

    for sentence_idx in range(len(post['sentences'])):

        s = post['sentences'][sentence_idx]
        for (term, _) in post['entity_interactions'][sentence_idx]:
            s = s.replace(term, '<strong>{0}</strong>'.format(term))

        post['markup'] += [s]

    filename = post['title'].replace("?", "") + '.entity_interactions.ht
    f = open(os.path.join(filename), 'wb')
    html = HTML_TEMPLATE.format(post['title'] + ' Interactions', ' '.joi
    f.write(html.encode('utf-8'))
    f.close()

    print('Data written to', f.name)

    # Display any of these files with an inline frame. This displays the
    # last file processed by using the last value of f.name...

    print('Displaying {0}:'.format(f.name))
    display(IFrame('files/{0}'.format(f.name), '100%', '600px'))
```

```
Data written to Four short links: 30 April 2020.entity_interactions.html
Displaying Four short links: 30 April 2020.entity_interactions.html:
```

To **Microservices** and **Back Again**: Why **Segment Went Back** to a **Monolith** — **microservices** came with increased operational **overhead** and **problems** around code reuse. **…** If **microservices** are implemented incorrectly or used as a **band-aid** without addressing some of the **root flaws** in your **system**, you'll be unable to do new **product development** because you're **drowning** in the complexity. **GNU** poke — interactive **editor** for binary data. Not limited to editing basic **entities** such as **bits** and **bytes**, it provides a full-fledged **procedural**, interactive **programming language** designed to describe data structures and to operate on them. (via **Kernel Recipes**) **Blender** — **Facebook** open sourced their **open-domain** ("can talk about **anything**!") chatbot. Human **evaluations** show our best **models** are superior to

```
In [ ]:
```