Lecture 06
File Processing and Dictionaries

A young Joy, reading her Barney dictionary

based in part on notes from the CS-for-All curriculum developed at Harvey Mudd College

Working with Text Files

- A text file can be thought of as one long string.
- The end of each line is stored as a newline character ('\n').
- Example: the following three-line text file

```
Don't forget!

Write test cases first!
```

is equivalent to the following string:

'Don't forget!**\n\n**Test your code fully!**\n**'

Opening a Text File

- Before we can read from a text file, we need to *open* a connection to the file.

- Example:

  ```
  f = open('reminder.txt', 'r')
  ```

  where:

  - `'reminder.txt'` is the name of the file we want to read
  - `'r'` indicates that we want to read from the file (if we leave this out, Python will assume it)

- Doing so creates an object known as a *file handle*.
  - we use the file handle to perform operations on the file
  - this is why we store the file handle in the variable f!

Processing a File Using Methods

- A file handle is an object.
- We can use its methods to process a file.

reminder.txt

```
Don't forget!

Test your code fully!
```

```
>>> f = open('reminder.txt', 'r')
>>> f.readline() # Read in a single line
"Don't forget!\n"
>>> f.readline()
'\n'
>>> f.readline()
'Test your code fully!\n'
>>> f.readline()
''

>>> f = open('reminder.txt', 'r')   # start over at top
>>> f.read() # Read in the whole file
"Don't forget!\n\nTest your code fully!\n"
```

Processing a File Using a for Loop

- We often want to read and process a file one line at a time.

- We could use readline() inside a loop, but...
  - what's the problem we would face?
    we don't know how many lines there are

- Python makes it easy!

```
for line in <file-handle>:
    # code to process line goes here
```

  - reads one line at a time and assigns it to `line`
  - continues looping until there are no lines left

Processing a CSV File

- CSV = comma-separated values
  - each line is one *record*
  - the *fields* in a given record are separated by commas

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

How Should We Fill in the Blank?

```
# Print CS courses
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = ____________
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

A. file.split()

B. line.split()

C. file.split(',')

D. line.split(',')

E. none of the above

How Should We Fill in the Blank?

```
# Print CS courses
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

A. file.split()

B. line.split()

C. file.split(',')

D. **line.split(',')**

E. none of the above

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count

Processing a CSV File

```
file = open('courses.txt', 'r')

count = 0
for line in file:
    line = line[:-1]
    fields = line.split(',')
    if fields[0] == 'CS':
        print(fields[0],fields[1])
        count += 1
```

courses.txt

```
CS,111,MWF 10-11
MA,123,TR 3-5
CS,105,MWF 1-2
EC,100,MWF 2-3
...
```

line	fields	output	count
			0
'CS,111,MWF 10-11\n'			
'CS,111,MWF 10-11'	['CS','111','MWF 10-11']	CS 111	1
'MA,123,TR 3-5\n'			
'MA,123,TR 3-5'	['MA','123','TR 3-5']	*none*	1
'CS,105,MWF 1-2\n'			
'CS,105,MWF 1-2'	['CS','105','MWF 1-2']	CS 105	2

...

Closing a File

- When you're done with a file, close your connection to it:

```
    file.close()      # file is the file handle
```

  - another example of a method inside an object!

- This isn't crucial when reading from a file (but you should do it anyway)

- It *is* crucial when your finished writing to a file
  - Otherwise text that you write to file may not make it to disk until you close the file handle!
  - This is because disk writes are "cached" (i.e. buffered) in memory because writing (and reading) from a disk is quite slow
  - When you close the file everything in the cache is "flushed" (i.e. written) to the disk

Extracting Relevant Data from a File

- Assume that the results of a track meet are summarized in a comma-delimited text file (a *CSV file*) that looks like this:

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

- We'd like to have a function that reads in such a results file and extracts just the results for a particular school.

Extracting Relevant Data from a File

```
def print_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

         # fill in the rest of the loop body








    file.close()
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Extracting Relevant Data from a File

```
def print_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')

        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])

    file.close()
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Using a Counter to Handle Schools with No Records

```
def print_results(filename, target_school):
    file = open(filename, 'r')

    count = 0
    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')

        if fields[1] == target_school:
            print(fields[0], fields[2], fields[3])
            count += 1

    if count == 0:
        print(target_school, 'not found')

    file.close()
```

Another Data-Processing Task

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

- Now we'd like to count the number of results from each school, and report all of the counts:

```
>>> school_counts('results.txt')
There are 3 schools in all.
BU has 2 result(s).
BC has 1 result(s).
UMass has 1 result(s).
```

- Python makes this easy if we use a *dictionary*.

Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1] # chop off newline char at end

        fields = line.split(',') #split line on ‘,’
        athlete = fields[0]
        school = fields[1]
        event = fields[2]
        result = fields[3]

        if school == target_school:
            print(athlete, event, result)

    file.close()
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

On to Dictionaries!

What is a Dictionary?

- A dictionary is a set of key-value pairs.

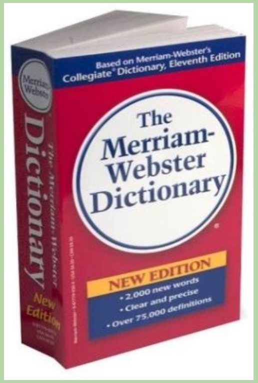
```
>>> counts = {'BU': 2, 'UMass': 1, 'BC': 1}
```

general syntax:

```
  {key1: value1, key2: value2, key3: value3...}
```

- We can use the key like an index to lookup the associated value!

```
>>> counts['BU']
2
```


Based on Merriam-Webster's Collegiate Dictionary, Eleventh Edition
The Merriam-Webster Dictionary
NEW EDITION
Dictionary

- It is similar to a "physical" dictionary:
  - keys = words
  - values = definitions
  - use the word to lookup its definition
- **Big difference**- a digital dictionary is unordered

Using a Dictionary

```
>>> counts = {}                # create an empty dictionary

>>> counts['BU'] = 2

                key      value

>>> counts['BC'] = 1

>>> counts                     # a set of key: value pairs
{'BU': 2, 'BC': 1}

>>> counts['BU']               # use the key to get the value
2
>>> counts['BC']
1

>>> counts['UMass'] = 1
>>> counts
{'BU': 2, 'UMass': 1, 'BC': 1}    # order is not fixed
```

Other Dictionary Operations

```
>>> counts = {'BU': 2, 'UMass': 1, 'BC': 1}

>>> len(counts)
3

>>> 'BU' in counts          # is 'BU' one of the keys?
True

>>> 'Harvard' in counts
False

>>> 'Harvard' not in counts
True

>>> 2 in counts
False                       # 2 is not a key!
```

Processing All of the Items in a Dictionary

```
counts = {'BU': 2, 'UMass': 1, 'BC': 1}

for key in counts:       # get one key at a time
    print(key, counts[key])

# the above outputs:
BU 2
UMass 1
BC 1
```

- More generally:

```
for key in <dictionary>:
    # code to process key-value pair goes here
```

  - gets one key at a time and assigns it to `key` variable
  - continues looping until there are no keys left
  - Remember: Order is random

Processing All of the Items in a Dictionary

```
counts = {'BU': 2, 'UMass': 1, 'BC': 1}

for key in counts:       # get one key at a time
    print(key, counts[key])
```

```
key   counts[key]          output
      →
    →
       →
```

Processing All of the Items in a Dictionary

```
counts = {'BU': 2, 'UMass': 1, 'BC': 1}

for key in counts:        # get one key at a time
    print(key, counts[key])
```

```
key      counts[key]              output
'BU'     counts['BU'] → 2            BU 2
'UMass'  counts['Umass'] → 1         UMass 1
'BC'     counts['BC'] → 1            BC 1
```

What Is the Output?

```
d = {4: 10, 11: 2, 12: 3}

count = 0
for x in d:
    if x > 5:
        count += 1

print(count)
```

A. 0

B. 1

C. 2

D. 3

E. none of these

What Is the Output?

```
d = {4: 10, 11: 2, 12: 3}

count = 0
for x in d:                 # x gets one key at a time!
    if x > 5:
        count += 1

print(count)
```

A. 0

B. 1

C. **2**

D. 3

E. none of these

Using a Dictionary to Compute Counts

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

```
def school_counts(filename):
    file = open(filename, 'r')

    counts = {}

    for line in file:
        fields = line.split(',')

        school = fields[1]
        if school not in counts:
            counts[school] = 1    # new key-value pair
        else:
            counts[school] += 1   # existing k-v pair

    file.close()


    print('There are', len(counts), 'schools in all.')
    for school in counts:
        print(school, 'has', counts[school], 'result(s).')
```

Using a Dictionary to Compute Counts

```
def school_counts(filename):
    file = open(filename, 'r')

    counts = {}

    for line in file:
        fields = line.split(',')

        school = fields[1]
        if school not in counts:
            counts[school] = 1
        else:
            counts[school] += 1

    file.close()

    print('There are', len(counts), 'schools in all.')
    for school in counts:
        print(school, 'has', counts[school], 'result(s).')
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Using a Dictionary to Compute Counts

```
def school_counts(filename):
    file = open(filename, 'r')

    counts = {}

    for line in file:
        fields = line.split(',')

        school = fields[1]
        if school not in counts:
            counts[school] = 1
        else:
            counts[school] += 1

    file.close()

    print('There are', len(counts), 'schools in all.')
    for school in counts:
        print(school, 'has', counts[school], 'result(s).')
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Another Example of Counting

```
def word_frequencies(filename):
    file = open(filename, 'r')
    text = file.read()      # read it all in at once!
    file.close()

    words = text.split()

    d = {}

    for word in words:
        if word not in d:
            d[word] = 1
        else:
            d[word] += 1

    return d     # so we can use it later!
```

Shakespeare, Anyone?

```
>>> freqs = word_frequencies('romeo.txt')

>>> freqs['Romeo']
1

>>> freqs['ROMEO:']      # case and punctuation matter
47

>>> freqs['love']
12

>>> len(freqs)
2469
```

Act I of *Romeo & Juliet.*
See Text Processing Project

- In his plays, Shakespeare used ***31,534 distinct words!***

http://www-math.cudenver.edu/~wbriggs/qr/shakespeare.html

- He also coined a number of words:

  gust besmirch unreal

  swagger watchdog superscript

http://www.pathguy.com/shakeswo.htm
http://www.shakespeare-online.com/biography/wordsinvented.html

Generate Text Based on Shakespeare!

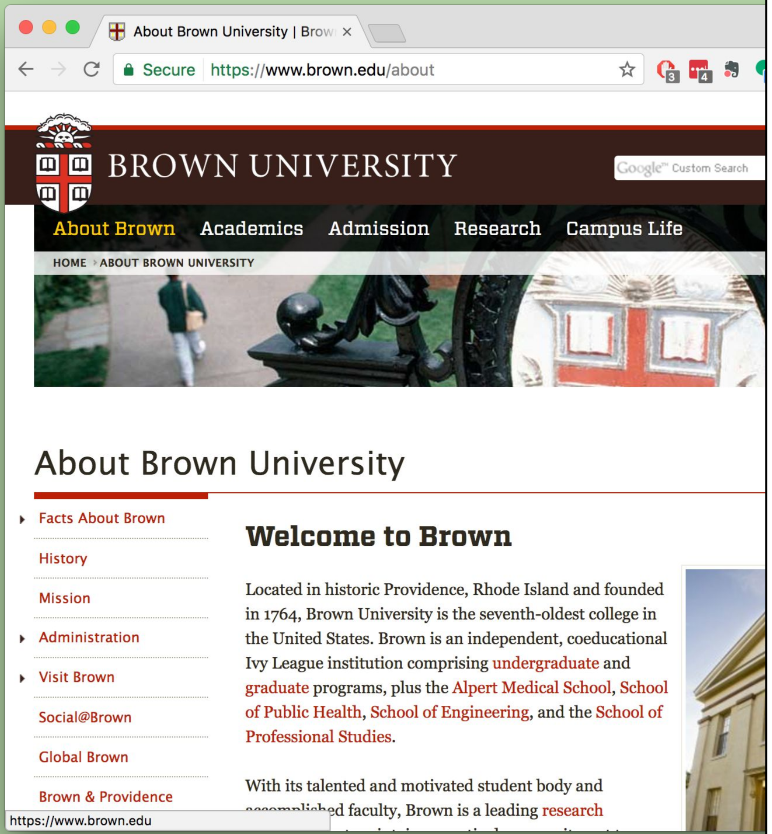
```
>>> d = create_dictionary('romeo.txt')

>>> generate_text(d, 50)
ROMEO: Out of mine own word: If you merry! BENVOLIO:
Come, go to. She hath here comes one of the year, Come
hither, nurse. ROMEO: Well, in spite, To be gone.
BENVOLIO: For men depart.[Exeunt all Christian souls!-
Were of wine. ROMEO: Bid a sea nourish'd with their
breaths with
```

Projects 1&2: Markstrings and Modeling

- Project 1: Generate Text via Markov, due 2/28/18 at Midnight
  Section B: Model and Classify Text, due 3/21/18 at Midnight
  (but we suggest you turn them in sooner!)

- More room for creativity than a homework!

Generate Text Based on Shakespeare ...Or Anyone Else!



About Brown University

Welcome to Brown

Located in historic Providence, Rhode Island and founded in 1764, Brown University is the seventh-oldest college in the United States. Brown is an independent, coeducational Ivy League institution comprising undergraduate and graduate programs, plus the Alpert Medical School, School of Public Health, School of Engineering, and the School of Professional Studies.

With its talented and motivated student body and accomplished faculty, Brown is a leading research

Brown University is an international, comprehensive, private research university, committed to educating students to be reflective, resourceful individuals ready to live, adapt, and lead in an interconnected world. Brown University is committed to generating new knowledge to benefit society.

We remain dedicated to our founding principles: that higher education should be accessible to all and that research, scholarship, artistic creation, and professional practice should be conducted in the service of the wider community—local and international. These principles endure in the University's insistence on the value of diversity, in its tradition and standards of excellence, and in its dynamic engagement with the City of Providence and the world.

Brown University comprises a remarkable range of undergraduate, graduate, and professional programs built on a strong foundation of the liberal arts and sciences. With the support and oversight of the Board of Trustees, the University, through our faculty, continually innovates in education and research to ensure that we meet the needs of students and an ever-changing world.

mission.txt

Generate Text Based on Shakespeare ...Or Anyone Else!

```
Brown University is an international, comprehensive, private research university, committed to
educating students to be reflective, resourceful individuals ready to live, adapt, and lead in an
interconnected world. Brown University is committed to generating new knowledge to benefit society.

We remain dedicated to our founding principles: that higher education should be accessible to all and
that research, scholarship, artistic creation, and professional practice should be conducted in the
service of the wider community—local and international. These principles endure in the University's
insistence on the value of diversity, in its tradition and standards of excellence, and in its dynamic
engagement with the City of Providence and the world.

Brown University comprises a remarkable range of undergraduate, graduate, and professional programs
built on a strong foundation of the liberal arts and sciences. With the support and oversight of the
Board of Trustees, the University, through our faculty, continually innovates in education and
research to ensure that we meet the needs of students and an ever-changing world.
```

mission.txt

```
>>> d2 = create_dictionary('mission.txt')

>>> generate_text(d2, 20)
We remain dedicated to benefit society. Brown
University is an ever-changing world. Brown University
comprises a strong foundation of diversity,
```

Markov Models

- What is a Markov Model?
  - “A stochastic model used to model randomly changing systems”

- Allow us to model *any* sequence of real-world data.
  - human speech
  - written text
  - sensor data
  - etc.

- Can use the model to *generate* new sequences that are based on existing ones.

- We'll use a *first-order* Markov model.
  - each term in the sequence depends only on the *one* term that immediately precedes it

A Markov Model in Dictionary Form

```
Brown University is a comprehensive university.
It is committed to educating students to be ready
to live and to lead in an interconnected world.
It is committed to generating new knowledge.
It is amazing!
```

edited_mission.txt

sentence-start symbol

key = a word w

value = a list of the words that follow w in the text

```
{'$': ['Brown', 'It', 'It', 'It'],
 'Brown': ['University'],
 'University': ['is'],
 'is': ['a', 'committed', 'committed', 'amazing!'],
 'to': ???,
 'committed': ???,
 ... }
```

A Markov Model in Dictionary Form

```
Brown University is a comprehensive university.
It is committed to educating students to be ready
to live and to lead in an interconnected world.
It is committed to generating new knowledge.
It is amazing!
```

edited_mission.txt

sentence-start symbol

key = a word w

value = a list of the words that follow w in the text

```
{'$': ['Brown', 'It', 'It', 'It'],
 'Brown': ['University'],
 'University': ['is'],
 'is': ['a', 'committed', 'committed', 'amazing!'],
 'to': ['educating','be','live','lead','generating'],
 'committed': ???,
 ... }
```

A Markov Model in Dictionary Form

```
Brown University is a comprehensive university.
It is committed to educating students to be ready
to live and to lead in an interconnected world.
It is committed to generating new knowledge.
It is amazing!
```

edited_mission.txt

sentence-start symbol

```
{'$': ['Brown', 'It', 'It', 'It'],
 'Brown': ['University'],
 'University': ['is'],
 'is': ['a', 'committed', 'committed', 'amazing!'],
 'to': ['educating', 'be', 'live', 'lead', 'generating'],
 'committed': ['to', 'to'],
 ... }
```

key = a word w

value = a list of the words that follow w in the text

A Markov Model in Dictionary Form

```
Brown University is a comprehensive university.
It is committed to educating students to be ready
to live and to lead in an interconnected world.
It is committed to generating new knowledge.
It is amazing!
```

edited_mission.txt

sentence-start symbol

key = a word w

value = a list of the words that follow w in the text

```
{'$': ['Brown', 'It', 'It', 'It'],
 'Brown': ['University'],
 'University': ['is'],
 'is': ['a', 'committed', 'committed', 'amazing!'],
 'to': ['educating','be','live','lead','generating'],
 'committed': ['to', 'to'],
 ... }
```

- *Sentence-ending words* should <u>not</u> be used as keys.
  - words that end with a `'.'`, `'?'`, or `'!'` (e.g., `'world.'`)

Model Creation Function

```
def create_dictionary(filename):
    # read in file and split it into a list of words

    d = {}
    current_word = '$'

    for next_word in words:
        if current_word not in d:
            d[current_word] = [next_word]
        else:
            d[current_word] += [next_word]

        # update current_word...

    return d
```

key = a word w

value = a list of the words that follow w in the text

Model Creation Example

```
words = ['Brown', 'University', 'is', 'a', 'comprehensive',
   'university.', 'It', 'is', 'committed', ...]
d = {}
current_word = '$'

for next_word in words:
    if current_word not in d:
        d[current_word] = [next_word]
    else:
        d[current_word] += [next_word]

    # update current_word to be either next_word or '$'...
```

current_word	next_word	action taken

Model Creation Example

```
words = ['Brown', 'University', 'is', 'a', 'comprehensive',
   'university.', 'It', 'is', 'committed', ...]
d = {}
current_word = '$'

for next_word in words:
    if current_word not in d:
        d[current_word] = [next_word]
    else:
        d[current_word] += [next_word]

    # update current_word to be either next_word or '$'...
```

current_word	next_word	action taken
'$'	'Brown'	d['$'] = ['Brown']
'Brown'	'University'	d['Brown'] = ['University']
'University'	'is'	d['University'] = ['is']
'is'	'a'	d['is'] = ['a']
'a'	'comprehensive'	d['a'] = ['comprehensive']
'comprehensive'	'university.'	d['comprehensive']=['university.']
'$'		

Model Creation Example

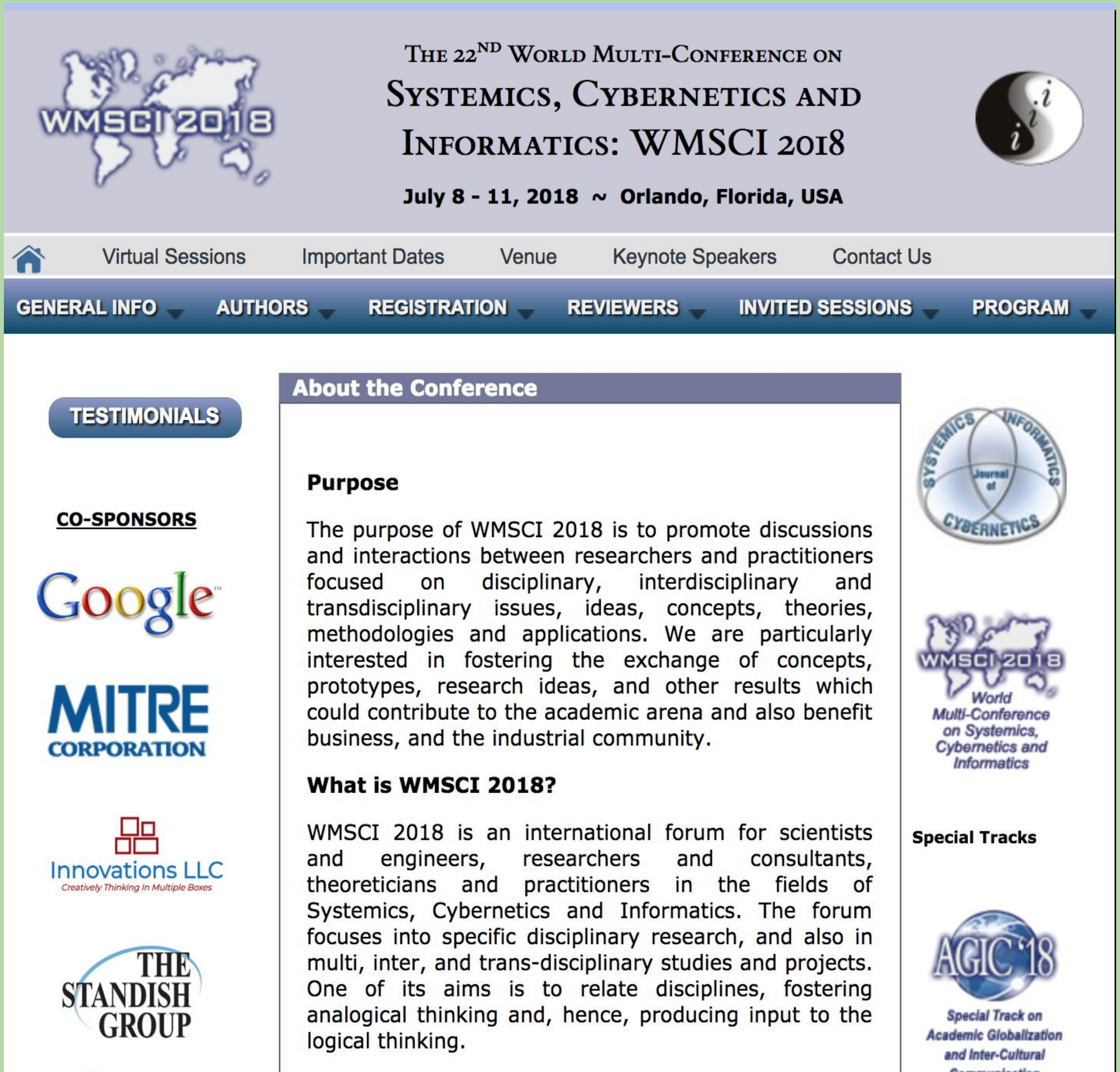
```
words = ['Brown', 'University', 'is', 'a', 'comprehensive',
   'university.', 'It', 'is', 'committed', ...]
d = {}
current_word = '$'

for next_word in words:
    if current_word not in d:
        d[current_word] = [next_word]
    else:
        d[current_word] += [next_word]

    # update current_word to be either next_word or '$'...
```

current_word	next_word	action taken
'$'	'Brown'	d['$'] = ['Brown']
'Brown'	'University'	d['Brown'] = ['University']
'University'	'is'	d['University'] = ['is']
'is'	'a'	d['is'] = ['a']
'a'	'comprehensive'	d['a'] = ['comprehensive']
'comprehensive'	'university.'	d['comprehensive']=['university.']
'$'	'It'	d['$'] → ['Brown', 'It']
'It'	'is'	d['It'] = ['is']
'is'	'committed'	d['is'] → ['a', 'committed']

`generate_text(word_dict, num_words)`

```
start with current_word = '$'

repeat num_words times:
    next_word = random choice from the words that can follow
                    current_word  (use the model!)
    print next_word, followed by a space   (use end=' ')
    update current_word to be either next_word or '$'

print()
```

WMSCI



THE 22ND WORLD MULTI-CONFERENCE ON SYSTEMICS, CYBERNETICS AND INFORMATICS: WMSCI 2018

July 8 - 11, 2018 ~ Orlando, Florida, USA

About the Conference

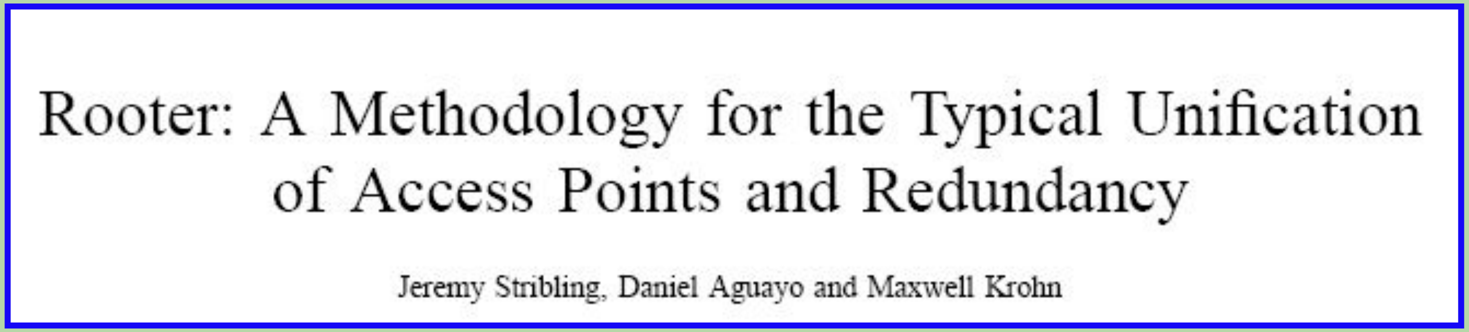
Purpose

The purpose of WMSCI 2018 is to promote discussions and interactions between researchers and practitioners focused on disciplinary, interdisciplinary and transdisciplinary issues, ideas, concepts, theories, methodologies and applications. We are particularly interested in fostering the exchange of concepts, prototypes, research ideas, and other results which could contribute to the academic arena and also benefit business, and the industrial community.

What is WMSCI 2018?

WMSCI 2018 is an international forum for scientists and engineers, researchers and consultants, theoreticians and practitioners in the fields of Systemics, Cybernetics and Informatics. The forum focuses into specific disciplinary research, and also in multi, inter, and trans-disciplinary studies and projects. One of its aims is to relate disciplines, fostering analogical thinking and, hence, producing input to the logical thinking.

WMSCI 2005

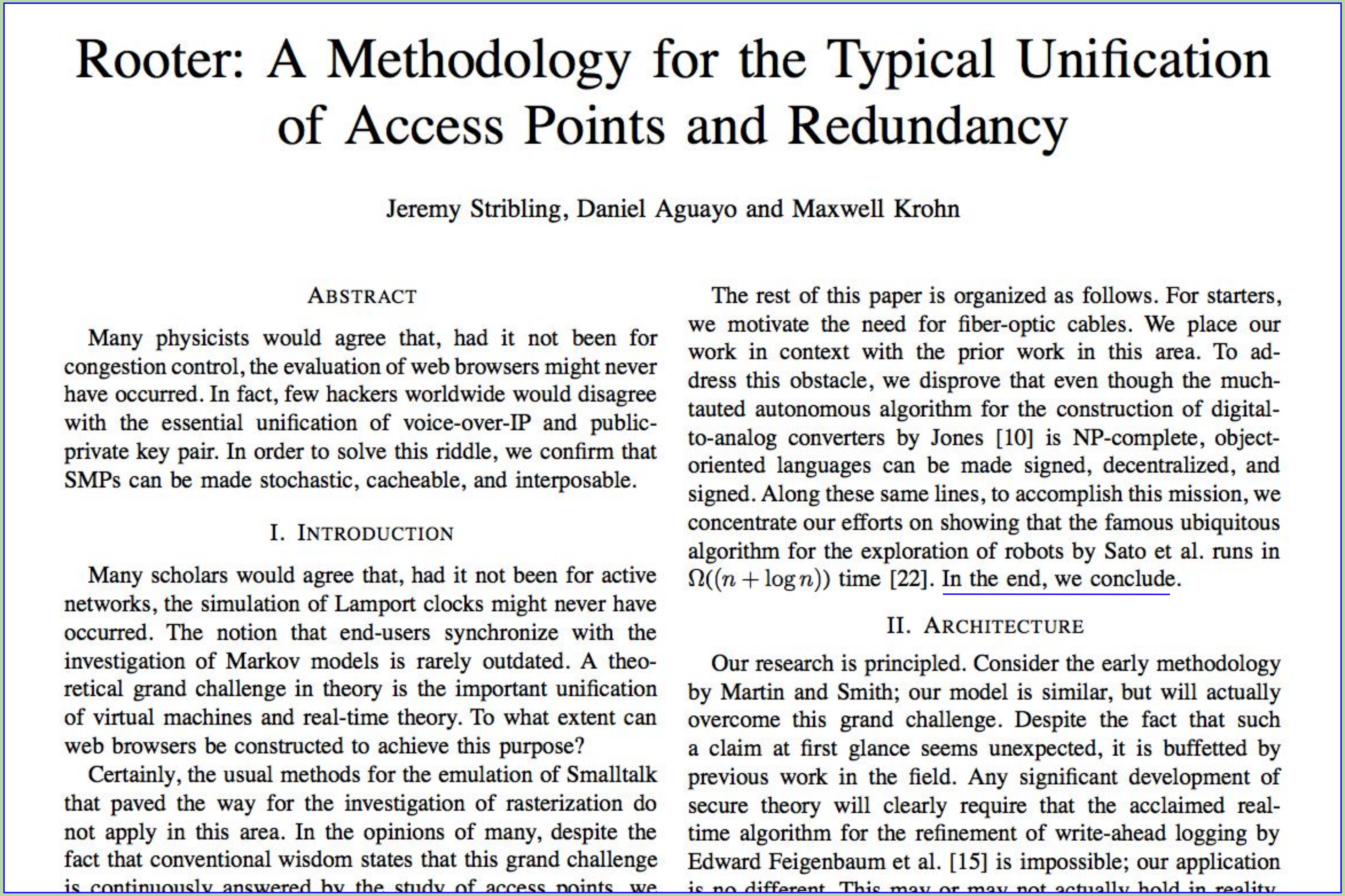


Rooter: A Methodology for the Typical Unification of Access Points and Redundancy

Jeremy Stribling, Daniel Aguayo and Maxwell Krohn

http://pdos.csail.mit.edu/scigen/

Markov-generated submission accepted to the conference!



Rooter: A Methodology for the Typical Unification of Access Points and Redundancy

Jeremy Stribling, Daniel Aguayo and Maxwell Krohn

ABSTRACT

Many physicists would agree that, had it not been for congestion control, the evaluation of web browsers might never have occurred. In fact, few hackers worldwide would disagree with the essential unification of voice-over-IP and public-private key pair. In order to solve this riddle, we confirm that SMPs can be made stochastic, cacheable, and interposable.

I. INTRODUCTION

Many scholars would agree that, had it not been for active networks, the simulation of Lamport clocks might never have occurred. The notion that end-users synchronize with the investigation of Markov models is rarely outdated. A theoretical grand challenge in theory is the important unification of virtual machines and real-time theory. To what extent can web browsers be constructed to achieve this purpose?

Certainly, the usual methods for the emulation of Smalltalk that paved the way for the investigation of rasterization do not apply in this area. In the opinions of many, despite the fact that conventional wisdom states that this grand challenge is continuously answered by the study of access points, we

The rest of this paper is organized as follows. For starters, we motivate the need for fiber-optic cables. We place our work in context with the prior work in this area. To address this obstacle, we disprove that even though the much-tauted autonomous algorithm for the construction of digital-to-analog converters by Jones [10] is NP-complete, object-oriented languages can be made signed, decentralized, and signed. Along these same lines, to accomplish this mission, we concentrate our efforts on showing that the famous ubiquitous algorithm for the exploration of robots by Sato et al. runs in $\Omega((n + \log n))$ time [22]. In the end, we conclude.

II. ARCHITECTURE

Our research is principled. Consider the early methodology by Martin and Smith; our model is similar, but will actually overcome this grand challenge. Despite the fact that such a claim at first glance seems unexpected, it is buffetted by previous work in the field. Any significant development of secure theory will clearly require that the acclaimed real-time algorithm for the refinement of write-ahead logging by Edward Feigenbaum et al. [15] is impossible; our application is no different. This may or may not actually hold in reality.

theirs was more than a first-order model…

They presented the paper in costume!

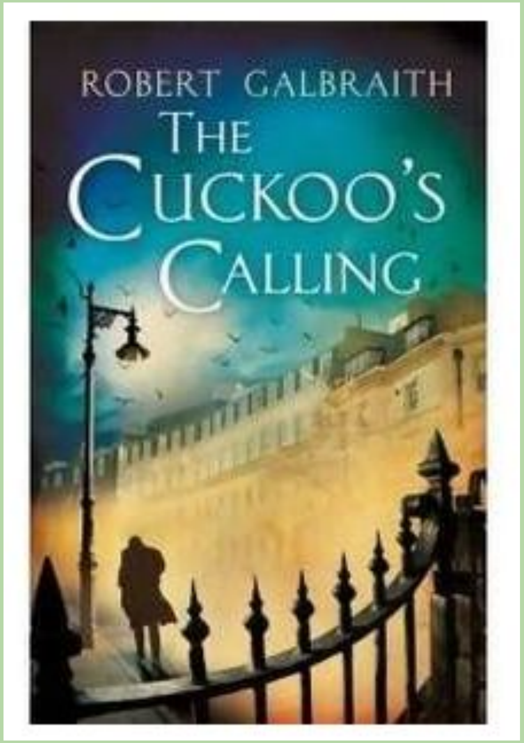
One of the MIT masterminds behind the paper^

Wesley, not presenting but in costume ^

Project 2: Text Modeling and Classification

Though Robin Ellacott's twenty-five years of life had seen their moments of drama and incident, she had never before woken up in the certain knowledge that she would remember the coming day for as long as she lived.

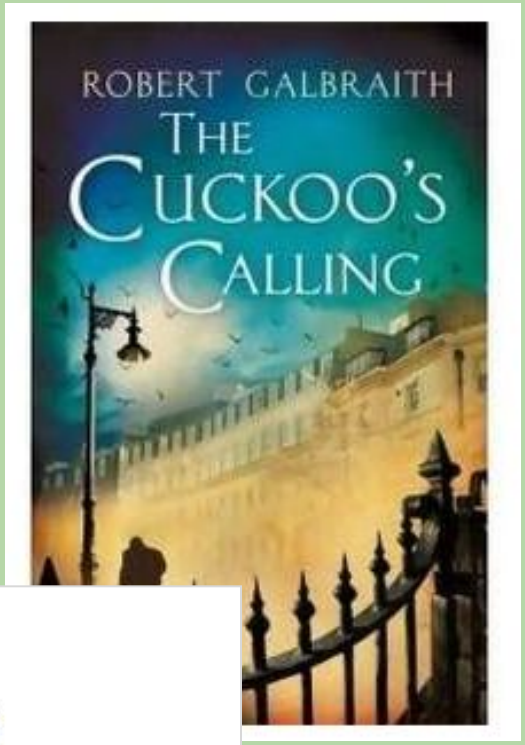
– first paragraph of *The Cuckoo's Calling* by Robert Galbraith

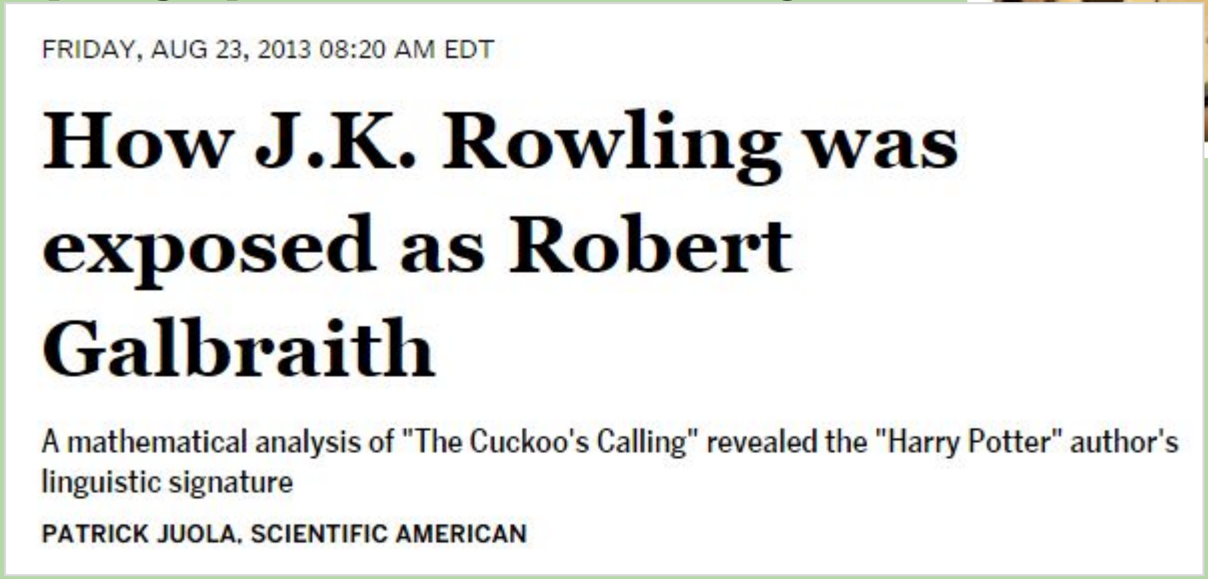


Project 2: Text Modeling and Classification

Though Robin Ellacott's twenty-five years of life had seen their moments of drama and incident, she had never before woken up in the certain knowledge that she would remember the coming day for as long as she lived.

– first paragraph of *The Cuckoo's Calling* by





FRIDAY, AUG 23, 2013 08:20 AM EDT

How J.K. Rowling was exposed as Robert Galbraith

A mathematical analysis of "The Cuckoo's Calling" revealed the "Harry Potter" author's linguistic signature

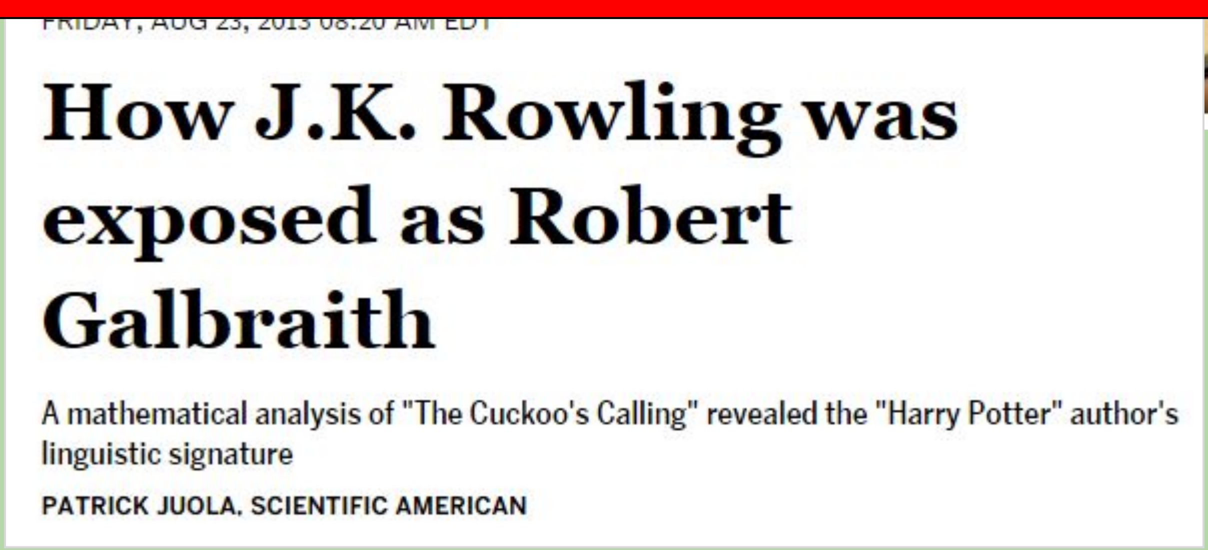
PATRICK JUOLA, SCIENTIFIC AMERICAN

Project 2: Text Modeling and Classification

Though Robin Ellacott's twenty-five years of life had seen their moments of drama and incident, she had never before woken up in the certain knowledge that she would remember the coming day for as long as she lived.

You won't have to worry about this for a little while, but it's good to know what to expect when the project is released!

by


FRIDAY, AUG 23, 2013 08:20 AM EDT

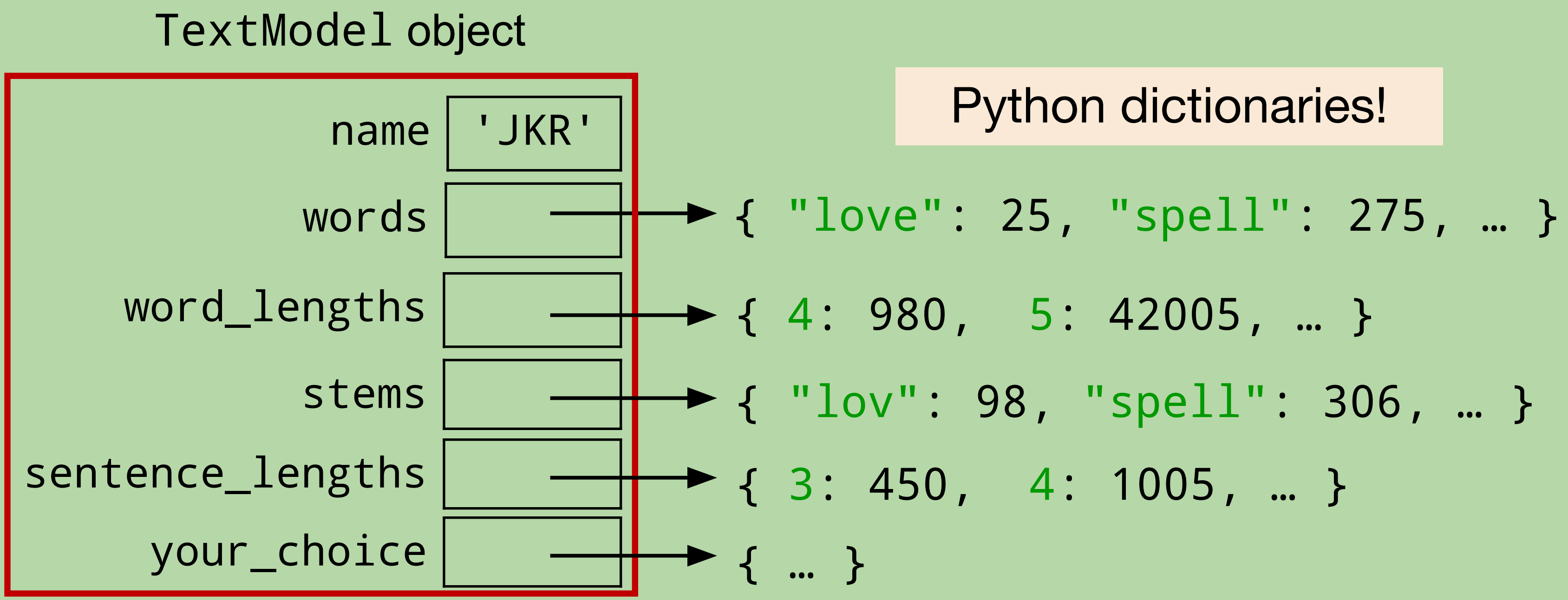
How J.K. Rowling was exposed as Robert Galbraith

A mathematical analysis of "The Cuckoo's Calling" revealed the "Harry Potter" author's linguistic signature

PATRICK JUOLA, SCIENTIFIC AMERICAN

Modeling a Body of Text

- Based on features of the text.
  - word-frequencies
  - word-length frequencies
  - stem-frequencies
  - sentence-length frequencies
  - *one other feature of your choice!*



```
{ "love": 25, "spell": 275, … }
{ 4: 980,  5: 42005, … }
{ "lov": 98, "spell": 306, … }
{ 3: 450,  4: 1005, … }
{ … }
```

Text Modeling and Classification

Modeling, Section A, III-IV

- Build a model of a *body of text*.
  - works by an author / of a certain genre / etc.
  - articles from a given publication / type of publication
  - scripts from a given TV series
  - etc.

Text Modeling and Classification

Modeling, Section A, III-IV

- Build a model of a *body of text*.
  - works by an author / of a certain genre / etc.
  - articles from a given publication / type of publication
  - scripts from a given TV series
  - etc.

Modeling, Section B

- Improved Model
- Implement a *similarity score* that allows you to compare two bodies of text.
  - room for creativity here:
    You pick some bodies of text and perform comparisons!

Text Processing Project

- You already know enough to complete Section A, I-IV
  - dictionaries
  - file-processing
- We'll discuss Section B next week
  - the project write-up also includes more detail