Dive into natural language processing with Python and NLTK

Andrii Soldatenko
@a_soldatenko

Agenda:

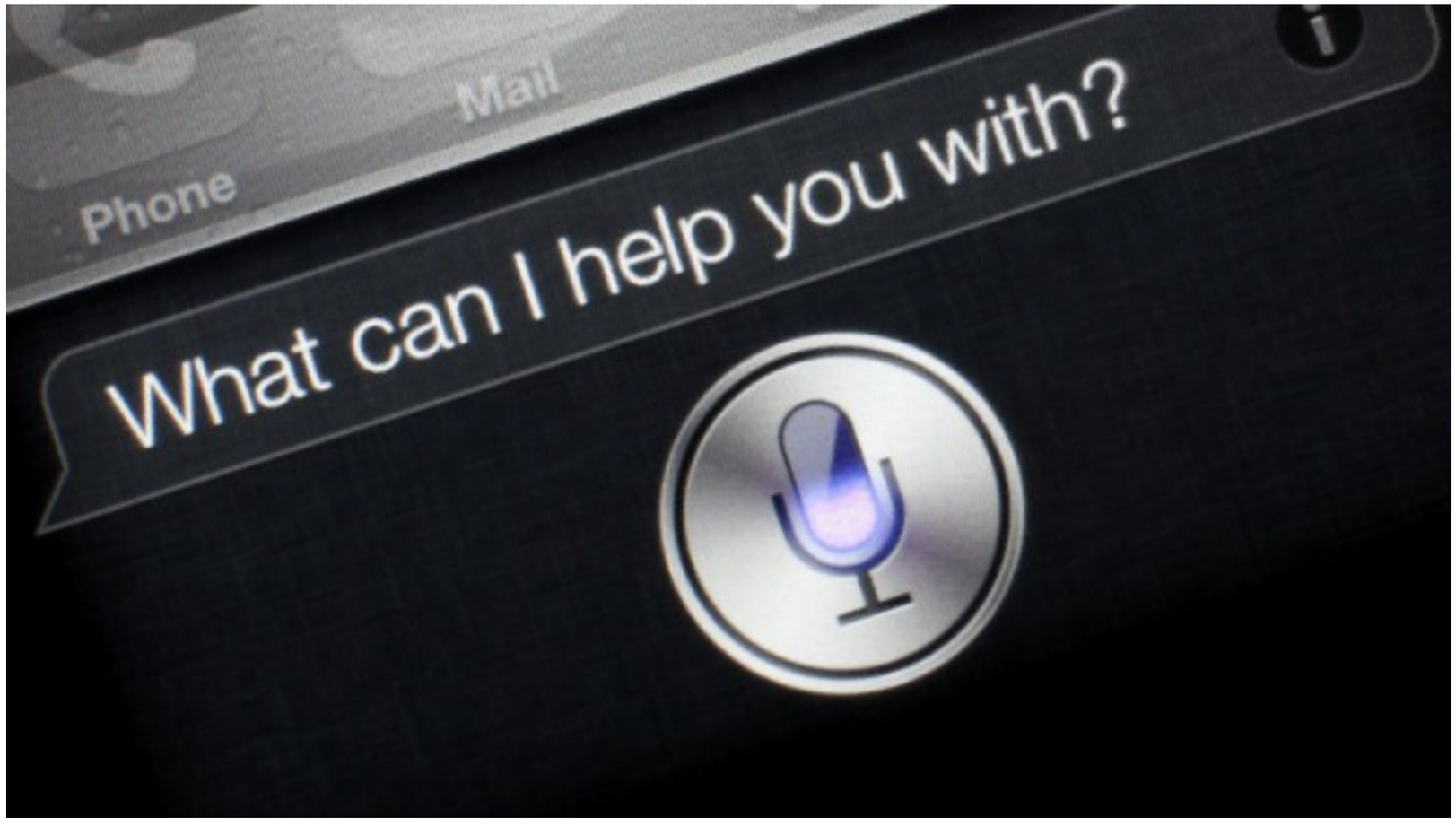
- Who am I?
- What is Natural language processing?
- Natural Language Toolkit (NLTK)
- Named-entity recognition (NER)
- Search speed
- What’s next?

Andrii Soldatenko

- Senior Python Developer at 
- CTO in Ethoos.com 
- Speaker at many PyCons and Python meetups
- blogger at https://asoldatenko.com

Preface

Siri



Text Search

➜ cpython time grep -r -i 'OrderedDict' .

grep -r -i 'OrderedDict' **2.35s user 0.10s system 97% cpu 2.510 total**

➜ cpython time ack OrderedDict

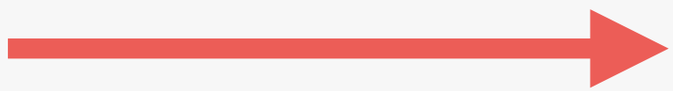
ack OrderedDict **1.74s user 0.14s system 96% cpu 1.946 total**

➜ cpython time pss OrderedDict

pss OrderedDict **0.85s user 0.09s system 96% cpu 0.983 total**

➜ **cpython time pt OrderedDict**

pt OrderedDict 0.14s user 0.10s system 462% cpu 0.051 total



Processor 2.5 GHz Intel Core i7

Memory 16 GB 1600 MHz DDR3

Natural Language Toolkit

NLTK: Crash Course

Natural Language Toolkit
REST Api

```
➜  curl -d "text=python" \
> http://text-processing.com/api/stem/
{
    "text": "python"
}
```

Tokenization

Tokenization: Example

```
>>> from nltk.tokenize import TreebankWordTokenizer
>>> tokenizer = TreebankWordTokenizer()

>>> tokenizer.tokenize('Software engineering
conference’)

['Software', 'engineering', 'conference']
```

Tokenization: Oops :(

```
>>> from nltk.tokenize import
TreebankWordTokenizer
>>> tokenizer = TreebankWordTokenizer()
>>> tokenizer.tokenize("Can't tokenize
this")
['Ca', "n't", 'tokenize', 'this']
```

Tokenization: Oops :(2

```
>>> from nltk.tokenize import
WordPunctTokenizer
>>> tokenizer = WordPunctTokenizer()
>>> tokenizer.tokenize("Can't tokenize
this")
['Can', "'", 't', 'tokenize', 'this']
```

Tokenization: Regular Expression

```
>>> from nltk.tokenize import
RegexpTokenizer
>>> tokenizer = RegexpTokenizer("[\w']+")
>>> tokenizer.tokenize("Can't tokenize
this")
["Can't", 'tokenize', 'this']
```

When your regular expression returns what you expect:)

WordNet - a lexical database for English language

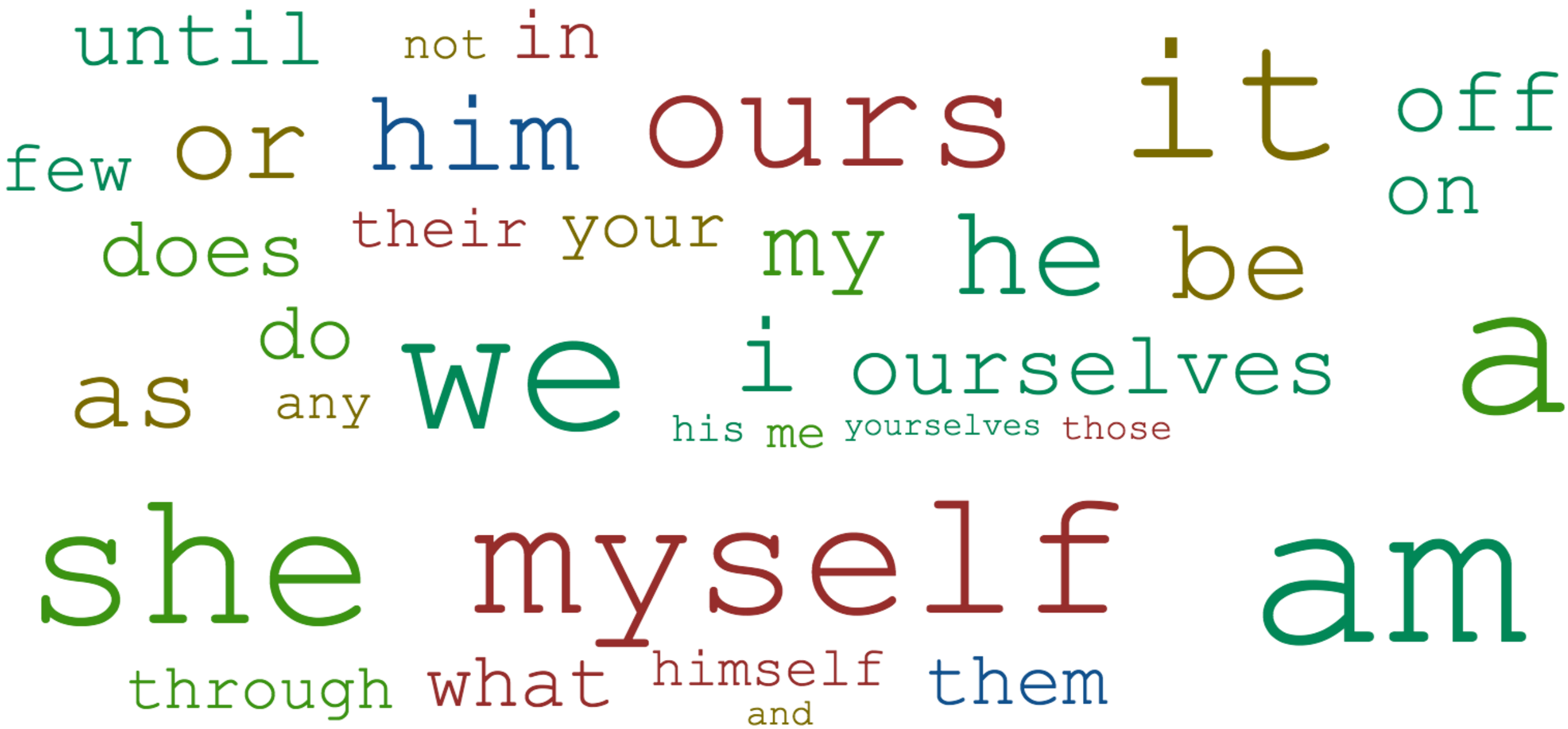
Download Data

```
➜  python -V
Python 3.5.2

➜  python -m nltk.downloader all
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /home/
vagrant/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package alpino to
...

➜  python -m nltk.downloader punkt
[nltk_data] Downloading package punkt to...
[nltk_data]   Unzipping tokenizers/punkt.zip
```

Stop Words



/home/vagrant/nltk_data/corpora/stopwords/english

StopWords: total count

```
➜ cat /home/vagrant/nltk_data/corpora/stopwords/english | wc -l
153

➜ cat /usr/share/postgresql/9.5/tsearch_data/english.stop | wc -l
127
```

StopWords: lang count

```
➜ ls /home/vagrant/nltk_data/corpora/stopwords/ | grep -v README | wc -l
15
```

StopWords: diff

```
➜ diff /home/vagrant/nltk_data/corpora/stopwords/english /usr/share/postgresql/9.5/tsearch_data/english.stop
128,153d127
< d
< ll
< m
< o
< re
< ve
< y
< ain
< aren
```

StopWords: Example

```
>>> from nltk.corpus import stopwords
>>> english_stops =
set(stopwords.words('english'))
>>> words = [“Can’t”, “tokenize”, “this”]
>>> [word for word in words if word not in
english_stops]
["Can't tokenize"]
```

Improve Stop words

```
import re

replacement_patterns = [
    (r'won\'t', 'will not'),
    (r'can\'t', 'can not'),
    ...
]
```

```
class RegexpReplacer(object):
    def __init__(self, patterns=replacement_patterns):
        self.patterns = [(re.compile(regex), repl) for (regex,
repl) in
                         patterns]

    def replace(self, text):
        s = text
        for (pattern, repl) in self.patterns:
            s = re.sub(pattern, repl, s)
        return s
```

http://bit.ly/2bJFQUP

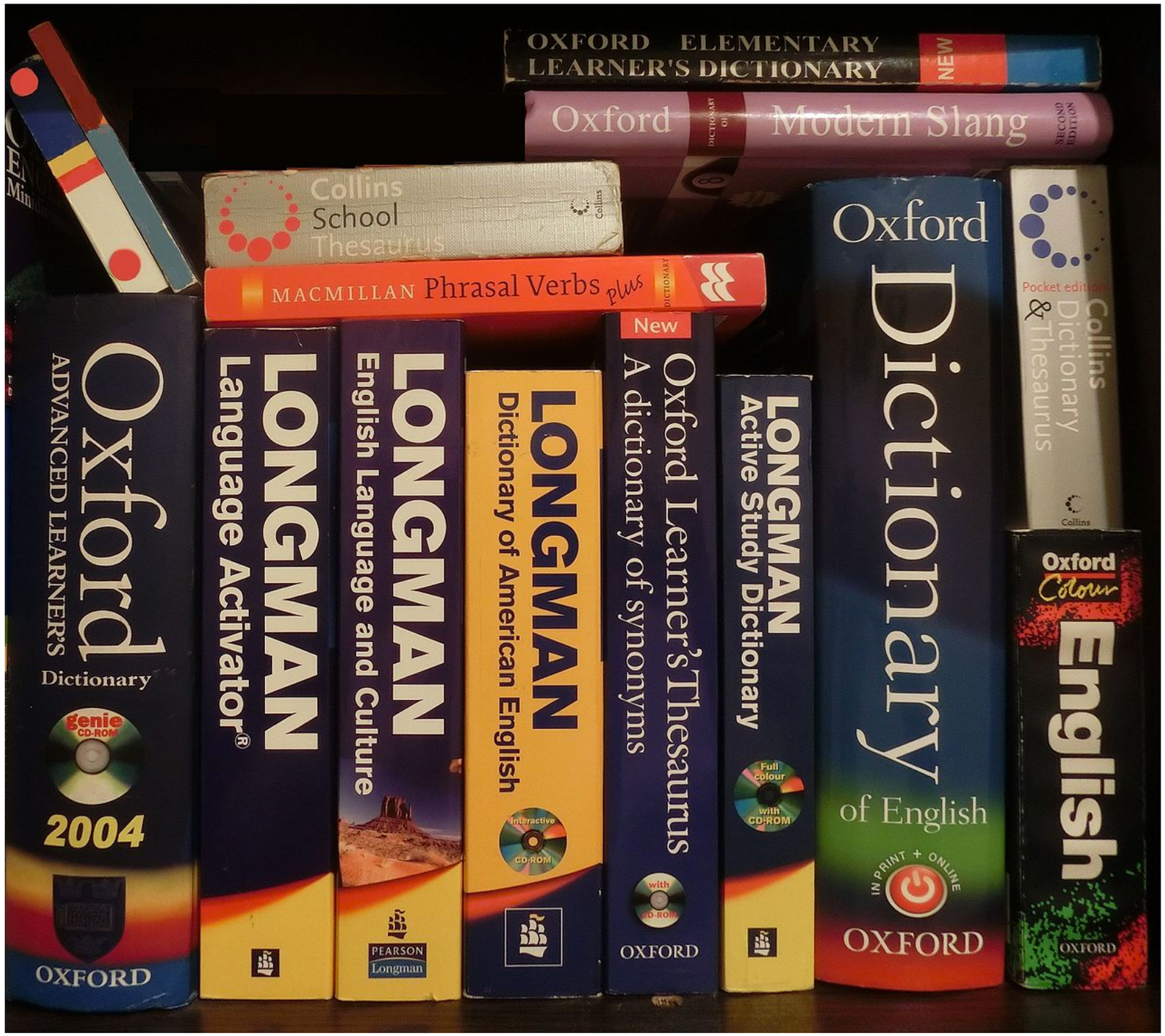
Improve Stop words

```
>>> english_stops = set(stopwords.words('english'))
>>> sentence = "Can't tokenize this"
>>> replacer = RegexpReplacer()
>>> tokenizer = WordPunctTokenizer()
>>>
>>> words =
tokenizer.tokenize(replacer.replace(sentence.lower()))
>>> [word for word in words if replacer.replace(word)
not in english_stops])
['tokenize']
```

Custom class

http://bit.ly/2bJFQUP

Collocations

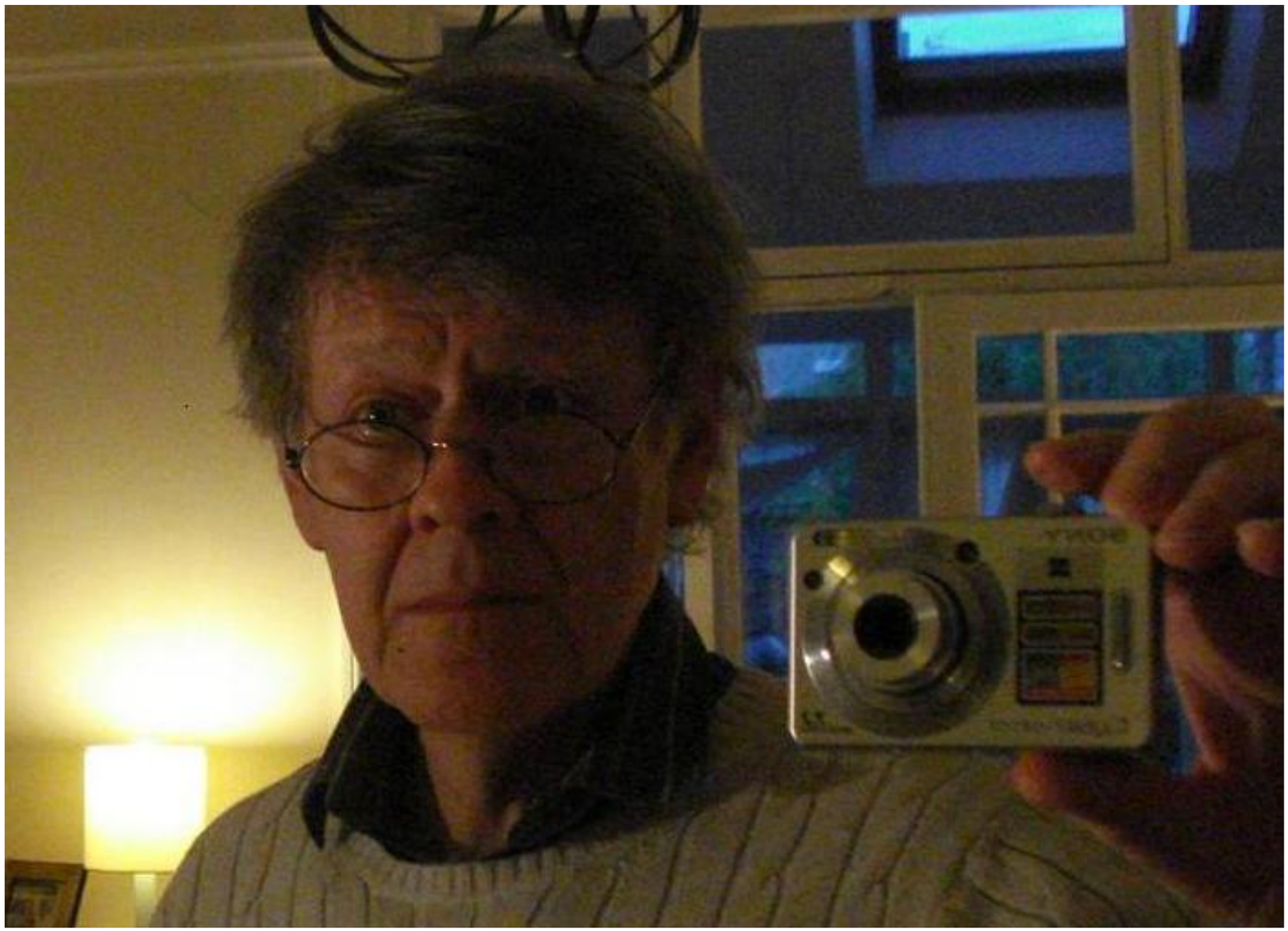


Collocations:Example

```
>>> words = [w.lower() for w in
webtext.words('grail.txt')]
>>> bcf = BigramCollocationFinder.from_words(words)
>>> results =
bcf.nbest(BigramAssocMeasures.likelihood_ratio, 10)
>>> pprint.pprint([' '.join(r) for r in results])
['black knight',
 'clop clop',
 'head knight',
 'mumble mumble',
 'squeak squeak',
 'saw saw',
 'holy grail',
 'run away',
 'french guard',
 'cartoon character']
```

Full example: http://bit.ly/2c1Rzkv

Stemming



https://tartarus.org/martin/PorterStemmer/def.txt

Porter stemming algorithm

```
>>> from nltk.stem import PorterStemmer
>>> stemmer = PorterStemmer()
>>> stemmer.stem('cooking')
'cook'
>>> stemmer.stem('cookery')
'cookeri
```

source code of algorithm: nltk.stem.porter.PorterStemmer

Lancaster stemming algorithm

```
>>> from nltk.stem import
LancasterStemmer
>>> stemmer = LancasterStemmer()
>>> stemmer.stem('cooking')
'cook'
>>> stemmer.stem('cookery')
'cookery'
```

source code: nltk.stem.lancaster.LancasterStemmer

Lemmatizing

Lemmatizing

```
>>> from nltk.stem import (
...     PorterStemmer,
...     WordNetLemmatizer
... )
>>>
>>> stemmer = PorterStemmer()
>>> stemmer.stem('believes')
'believ'
>>>
>>> lemmatizer = WordNetLemmatizer()
>>> lemmatizer.lemmatize('believes')
'belief'
```

Full text search

Search index

Simple sentences

1. The quick brown fox jumped over the lazy dog

2. Quick brown foxes leap over lazy dogs in summer

Inverted index

```
Term       Doc_1  Doc_2
-------------------------
Quick     |       |  X
The       |   X   |
brown     |   X   |  X
dog       |   X   |
dogs      |       |  X
fox       |   X   |
foxes     |       |  X
in        |       |  X
jumped    |   X   |
lazy      |   X   |  X
leap      |       |  X
over      |   X   |  X
quick     |   X   |
summer    |       |  X
the       |   X   |
-------------------------
```

Inverted index

```
1   Term        Doc_1  Doc_2
2   ---------------------------
3   brown    |    X    |  X
4   quick    |    X    |
5   ---------------------------
6   Total    |    2    |  1
```

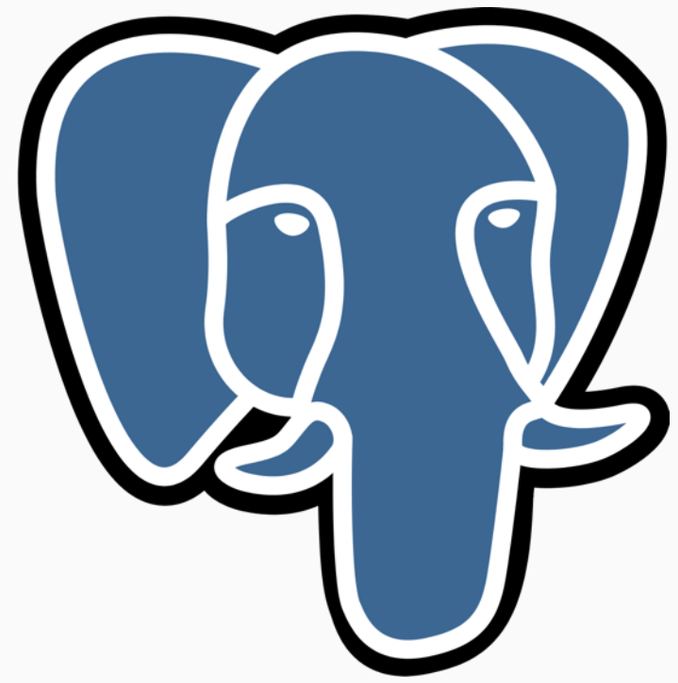
Inverted index: normalization

Term	Doc_1	Doc_2
Quick		X
The	X	
brown	X	X
dog	X	
dogs		X
fox	X	
foxes		X
in		X
jumped	X	
lazy	X	X
leap		X
over	X	X
quick	X	
summer		X
the	X	

Term	Doc_1	Doc_2
brown	X	X
dog	X	X
fox	X	X
in		X
jump	X	X
lazy	X	X
over	X	X
quick	X	X
summer		X
the	X	X

What is the best full text search engine in Python?

elastic



WHOOSH
Python search library

Sphinx

https://asoldatenko.com/europython-2016.html

NER

https://tartarus.org/martin/PorterStemmer/def.txt

EUROPYTHON 2016

Bilbao, 17-24 July

My talk about search engines you can find:

https://asoldatenko.com/europython-2016.html

1 million music Artists

Evie Tamala
Jean-Pierre Martin
Deejay One
wecamewithbrokenteeth
The Blackbelt Band
Giant Tomo
Decoding Jesus
Elvin Jones & Jimmy Garrison Sextet
Infester
...
David Silverman
Aili Teigmo

	Performance	Database size
elastic	9 ms	~ 1 million records
PostgreSQL	4 ms	~ 1 million records
Sphinx	6 ms	~ 1 million records
WHOOSH Python search library	~2 s	~ 1 million records

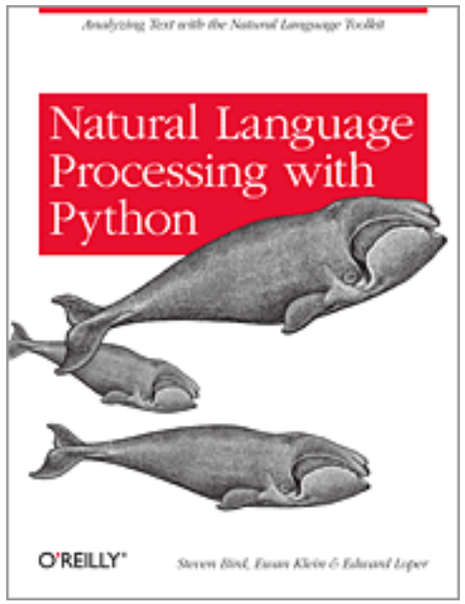
Named-entity recognition

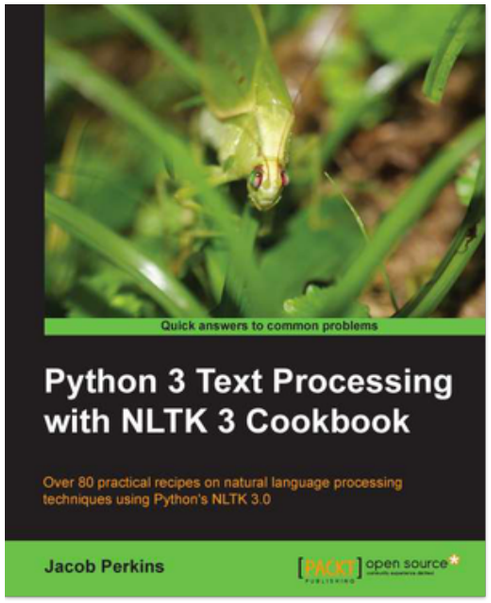
- Jim bought 300 shares of Acme Corp. in 2006.

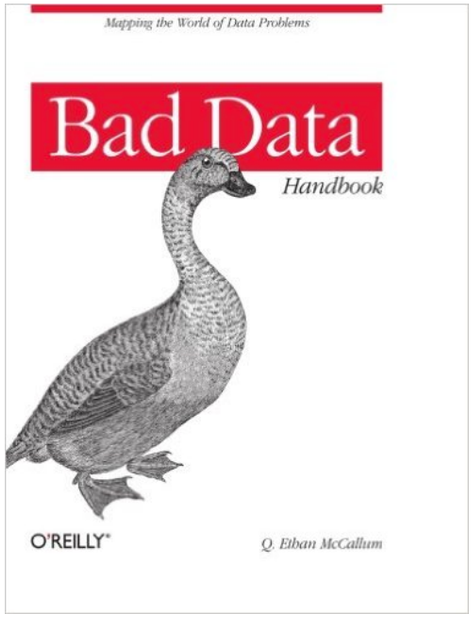
- [Jim]Person bought 300 shares of [Acme Corp.]Organization in [2006]Time.

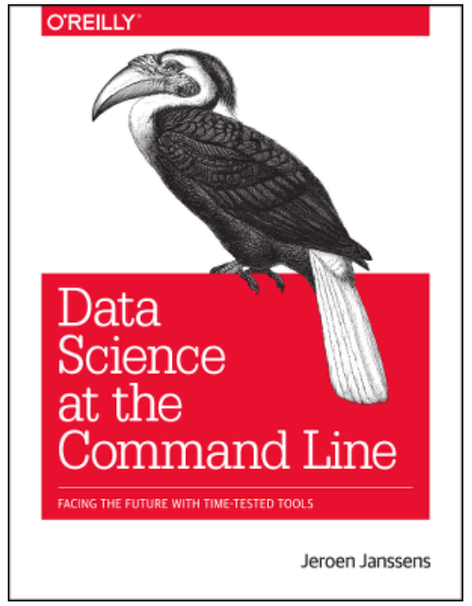
Books

Books

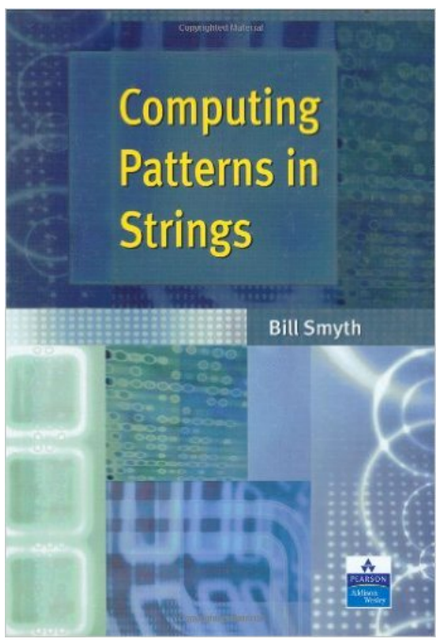

Analyzing Text with the Natural Language Toolkit
Natural Language Processing with Python
O'REILLY
Steven Bird, Ewan Klein & Edward Loper

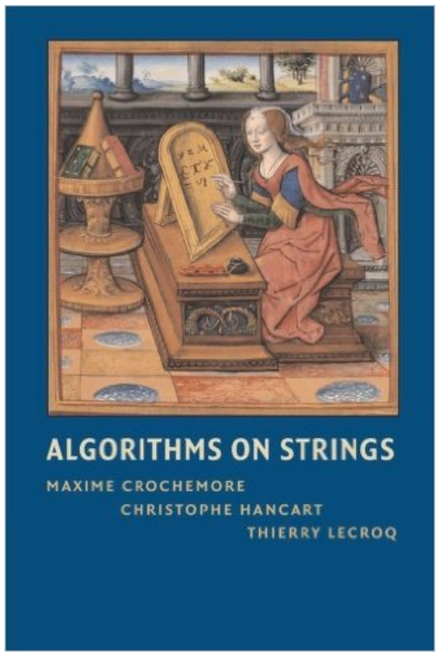

Quick answers to common problems
Python 3 Text Processing with NLTK 3 Cookbook
Over 80 practical recipes on natural language processing techniques using Python's NLTK 3.0
Jacob Perkins
[PACKT] open source

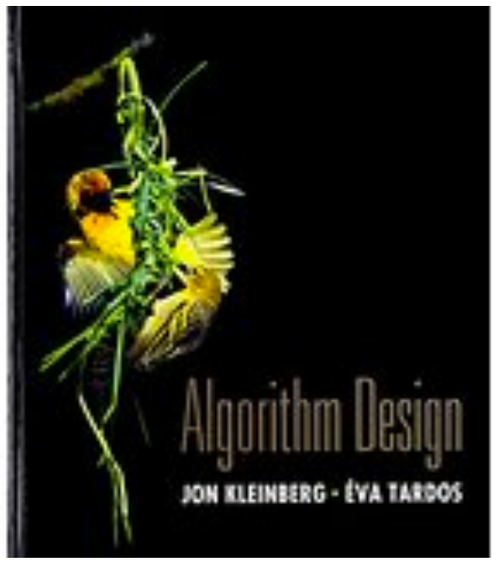

Mapping the World of Data Problems
Bad Data Handbook
O'REILLY
Q. Ethan McCallum


O'REILLY
Data Science at the Command Line
FACING THE FUTURE WITH TIME-TESTED TOOLS
Jeroen Janssens


Understand Your Data and Be More Productive
Mastering
Regular Expressions
O'REILLY
Jeffrey E.F. Friedl


Computing Patterns in Strings
Bill Smyth


ALGORITHMS ON STRINGS
MAXIME CROCHEMORE
CHRISTOPHE HANCART
THIERRY LECROQ


Algorithm Design
JON KLEINBERG · ÉVA TARDOS

References:

http://www.nltk.org/book_1ed/

http://bit.ly/2bKXnlz

Slides

https://asoldatenko.com/pdfs/se2016.pdf

http://bit.ly/2bJAuJb

Thank You

@a_soldatenko

andrii.soldatenko@gmail.com

/[Q|q]uestions/