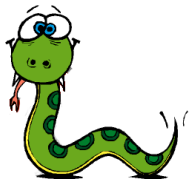


The Python & The Elephant

Large Scale Natural Language Processing with NLTK & Dumbo

Nitin Madnani
PhD Student
Department of Computer Science
Institute for Advanced Computer Studies
http://www.umiacs.umd.edu/~nmadnani
@haikuman

Jimmy Lin
Associate Professor
The iSchool
Institute for Advanced Computer Studies
http://www.umiacs.umd.edu/~jimmylin
@lintool

Presented at PyCon 2010
Atlanta, Georgia

Prologue

- Natural Language Processing (NLP) ⇔

  Computational Linguistics (CL) [this talk]
- Don't worry about grokking all the code now!
  [Everything's on the web and reproducible!]
- Might have to go a bit fast; Please hang on!
- Even if you don't care about NLP, please stick around for entertaining word association results
- Did I mention that my mom thinks I am really funny?

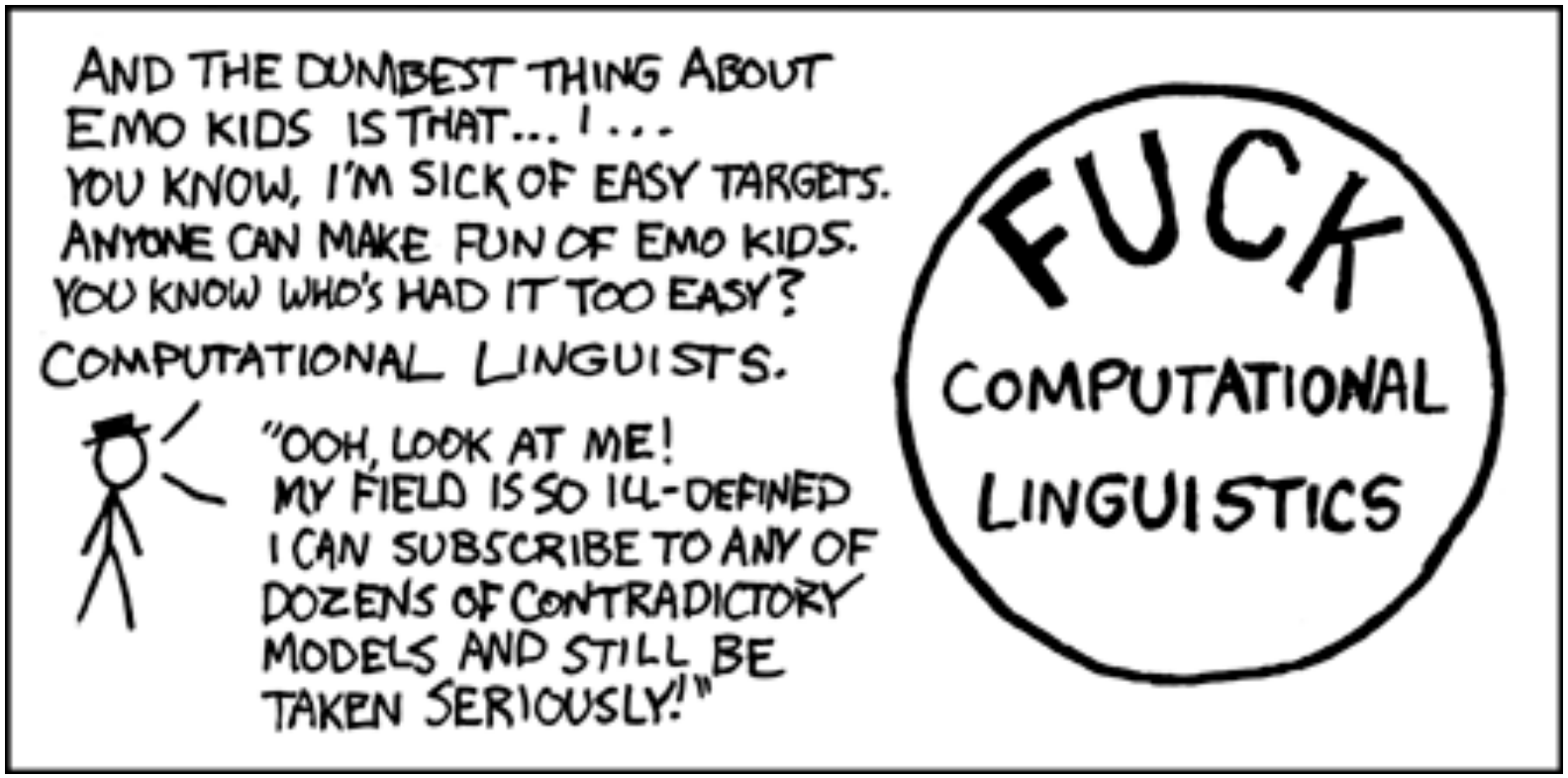
Introduction to NLP

- Extremely interdisciplinary field of study
  [Linguistics+Computer Science+Statistics]
- Some real world NLP problems:
  - Information Retrieval [Google/Bing/Yahoo!]
  - Statistical Machine Translation [Google Translate]
  - Entity and Relation Extraction ("Jimmy is Nitin's boss")
  - Automatic Text Summarization [Columbia Newsblaster]
  - Automatic Speech Recognition [Dragon NaturallySpeaking]

NLP is harder than it looks!

- Not every language has "words"!
  大江东流 ("the big river flows to the east")
- Ambiguities galore
  - Lexical ("bank" vs "bank")
  - Syntactic:
    - I saw the man with the telescope [Attachment]
    - I cooked her duck [Structure]
- Interesting (and *challenging*) area of research

Opinions on NLP vary though ...



Original version at http://xkcd.com/114/

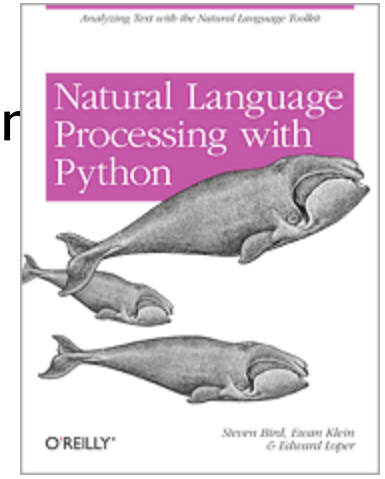
Python & NLP

- Python has:
  - Native unicode support
  - Extremely versatile standard library
  - Easy yet powerful text processing
  - Easily extensible using C/C++ (SWIG, Pyrex/Cython)
  - Low barrier to entry/Rapid prototyping
- Not *all* the "batteries" I want as an NLPer
- Enter NLTK: Open source Python NLP Toolkit

NLTK[1]: A Brief Introduction

A corpus is a body of written text.

- Fully self contained
  - Real-world data in the form of 50 *corpora* (raw & annotated)
  - Tokenizers, part-of-speech taggers, parsers, stemmers, machine learning tools etc.
- Integrated with WordNet[2], a database of semantic relationships for English nouns and verbs
  - Synonymy, Hyp(er/o)nymy [*is-a*], Holonymy/Meron
- Extremely active developer/user community
- The "Whale" book[3]



[1] http://www.nltk.org

[2] http://wordnet.princeton.edu/

[3] http://oreilly.com/catalog/9780596516499/

MapReduce in 120 seconds

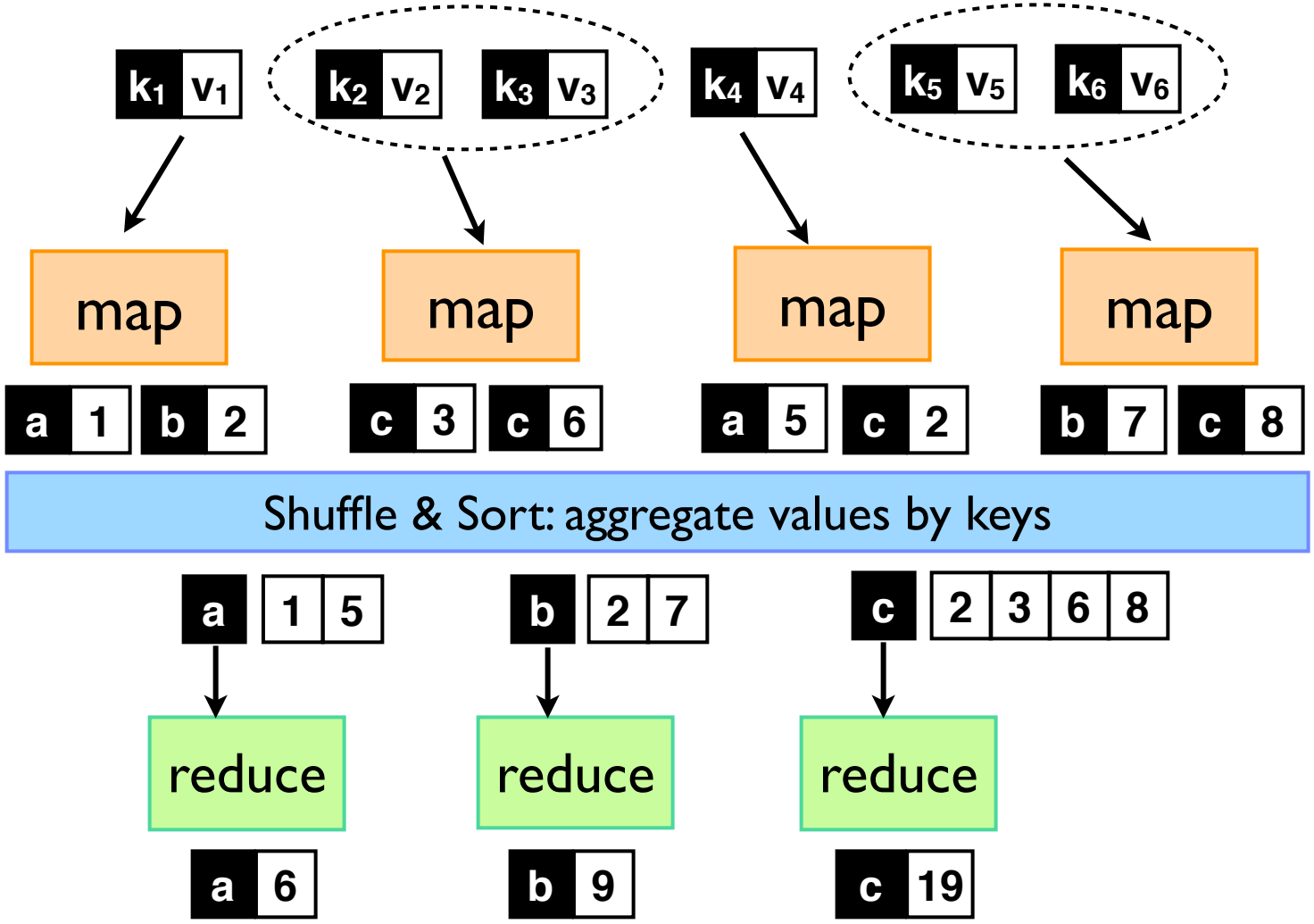
- Functional Programming + Distributed Processing
- *Every* datum is a key, value pair
  - Mappers & Reducers take (k, v)s and output (k,v)s
- *All* the user does is design the mapper and the reducer
- The execution framework handles *everything* else
  - Scheduling
  - Faults/Restarts
  - Synchronization ...

MapReduce in 120 seconds

- Functional Programming + Distributed Processing
- *Every* datum is a key, value pair
  - Mappers take (k, v)s and
  - Reducers ta (k, v)s
- *All* algorithms design the mapper and the reducer
- The execution framework handles *everything* else
  - Scheduling
  - Faults/Restarts
  - Synchronization ...

Screw this! Where's the picture?

MapReduce in Pictures



Counting Words

MapReduce Implementations

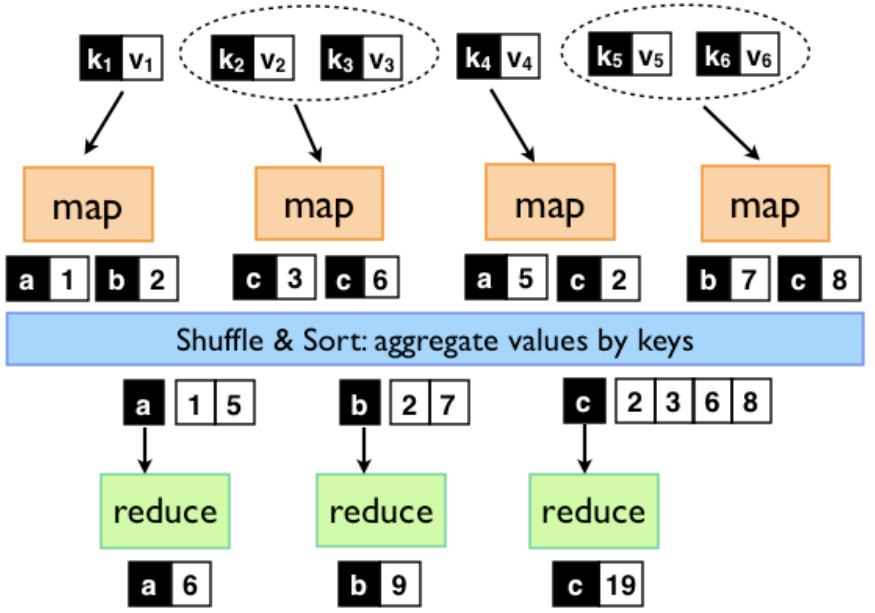
- MapReduce™: Google's C++ implementation
- Hadoop: Open-source Java implementation
- Dumbo: Python bindings for *Hadoop Streaming*
  - Use *any* executable/script as mapper and reducer
  - Allows using languages other than Java
  - Read/Write *lines* from/to Unix standard streams
  - Dumbo serializes to binary using `typedbytes`

Dumbo in Action

```
def mapper(k,v):
  for w in k.split(): yield w,1

def reducer(k,values):
  yield k,sum(values)

if __name__ == "__main__":
  import dumbo
  dumbo.run(mapper,reducer)
```



Counting Words

Does NLP need MapReduce?

- Google 5-gram corpus
  - 5-gram: sequence of five words
  - 1 trillion words, 24 GB compressed
- USENET corpus [Westbury Lab @ UAlberta]
  - Public USENET postings between Oct'05-Jan'10
  - 25 billion words, 28GB compressed
- ukWaC corpus [University of Bologna]
  - Web crawl of the .uk domain; tokenized and POS-tagged
  - 88 million sentences, 5.1GB compressed

A Pythonic Solution: NLTK + Dumbo

- Use NLTK for algorithms and data structures
- Use Dumbo for "Hadoopification"
- In this talk, I focus on the NLP task of word association
  - Common task in psycholinguistics in the context of *lexical retrieval, i.e., "what word Y (response) immediately comes to mind when hearing the word X (stimulus)?"*
  - NLP version: *Use a text corpus to figure out the word that occurs the most "near" the stimulus word*

Simple Word Association

- Work only with nouns
- Ignore non-content or "stop" words
- Rely on NLTK for data and data structures
  - The bundled POS-tagged **Brown** corpus
    [*Brown University Standard Corpus of "Present-Day" American English*]
  - The bundled stopwords corpus
  - The `ConditionalFreqDist()` data structure
    - Basically counts #(w_1|w_2), i.e., how many times was w_1 "associated with" w_2 in the corpus

Simple Word Association

```
from nltk.corpus import brown, stopwords
from nltk.probability import ConditionalFreqDist

cfd = ConditionalFreqDist()
stopwords_list = stopwords.words('english')

def is_noun(tag):
   return tag.startswith('N')

for sentence in brown.tagged_sents():
   for (index, tagtuple) in enumerate(sentence):
      (token, tag) = tagtuple
      token = token.lower()
      if token not in stopwords_list and is_noun(tag):
         window = sentence[index+1:index+5]
         for (window_token, window_tag) in window:
            window_token = window_token.lower()
            if window_token not in stopwords_list\
                  and is_noun(window_tag):
               cfd[token].inc(window_token)
```

('man', 'NN')

Import stuff and initialize CFD and stopword list

Count N_2 with N_1 if it occurs within 5 words of it

Nitin Madnani, *Getting Started on Natural Language Processing with Python*, ACM Crossroads, Volume 13, Issue 1, 2007.

Simple Word Association Results

```
print cfd['foo'].max()
```

the word "best associated" with 'foo'
(most frequently co-occurring)

Stimulus	Response
bread	butter
man	woman
life	death
tax	collection
hospital	admission
python	amethystine
web	earthmen(?)
justice	frankfurter

Nitin Madnani, *Getting Started on Natural Language Processing with Python*, ACM Crossroads, Volume 13, Issue 1, 2007.

Hadoopified Word Association

- Use Hadoop on an Amazon EC2 cluster
- Use ukWac corpus as input [freely available]
  - POS-tagged but **not** as clean (real-world)
  - More than just `is_noun()` and `not in stopwords_list`
- Rely on NLTK for `FreqDist()` data structure
  - Just counts words i.e., #(w_1)
  - Basically a fancy `dict()`
  - Hint: A CFD has a `FreqDist()` for each "condition"

Hadoopified Word Association

```
def mapper(key,value):
  sentence = value.split()
  for (i, tagtuple) in enumerate(sentence):
    token, tag = toktag(tagtuple)
      if we_like(token, tag):
        fd = FreqDist()
        token = token.lower()
        window = sentence[i+1:i+5]
        for windowtuple in window:
          wtoken, wtag = toktag(windowtuple)
          if we_like(wtoken, wtag):
            wtoken = wtoken.lower()
            fd.inc(wtoken)
       yield token, tuple(fd.items())
```

```
def reducer(key,values):
  finalfd = FreqDist()
  for fdstr in values:
    for k, v in fdstr:
      finalfd.inc(k, v)
  yield key, tuple(finalfd.items())
```

- Each mapper outputs a word and a list of tuples (`FreqDist()` ≈ a dictionary)
- For example, `('foo', [('bar', 5), ('baz', 4),..., ('zzz', 10)])`
- Each tuple list is a list of "*associated*" words & their "*strength of association*"
- Each reducer just combines all `FreqDist's` for its word into *one* `FreqDist()`
- We call this the ***Stripes*** MapReduce design pattern (a *stripe* for each word)

H-fied Word Association Results

Every reducer produces a text file for each word containing responses sorted by count

```
man     [woman, 7994],[god, 4457],[match, 3567],...,[zzkj, 1]]
```

Stimulus	Response
bread	butter
man	woman
life	death
tax	collection
hospital	admission
python	amethystine
web	earthmen
justice	frankfurter

Brown

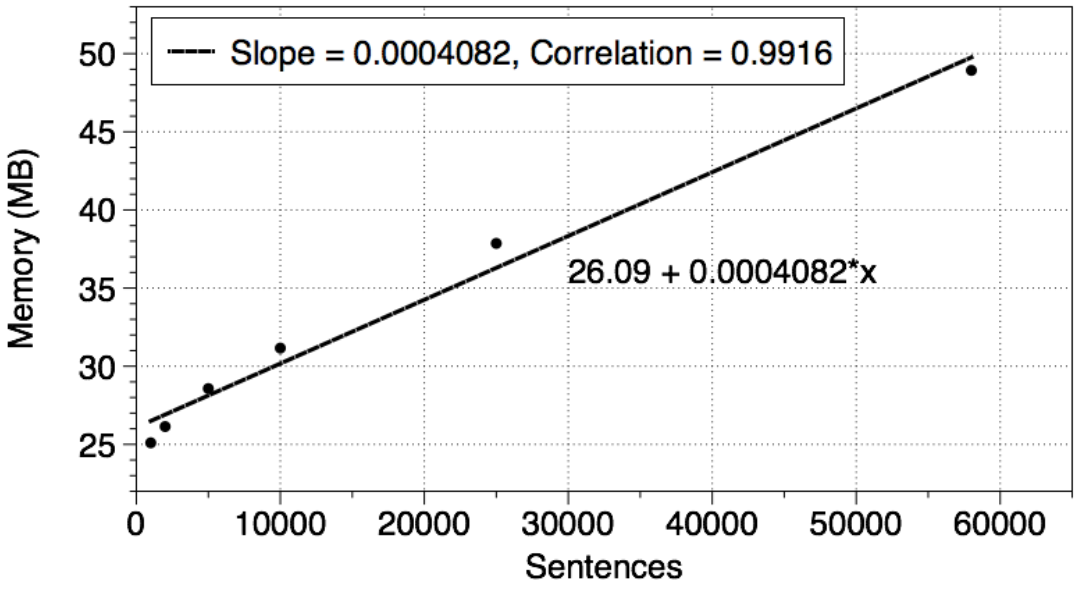
Stimulus	Response
bread	butter
man	woman
life	insurance
tax	credit
hospital	nhs
python	code
web	site
justice	system

ukWac

"holy"
"grail"

Did Hadoopifying help?

- What if we ran on 1 machine?
- Memory consumption: ~30GB
- Simple version takes 1 minute to process the entire Brown corpus (97,000 sentences)
- 88 million sentences will take ~15 hours on a single machine [assuming memory's okay(!)]



Estimated linear fit for memory size
VS
# of sentences in Brown corpus

Did Hadoopifying help?

domU-12-31-39-07-84-78 Hadoop Map/Reduce Administration

State: RUNNING
Started: Wed Feb 17 16:49:24 UTC 2010
Version: 0.20.2-dev, r${cloudera.hash}
Compiled: Fri Jan 8 14:40:54 EST 2010 by nmadnani
Identifier: 201002171649

Cluster Summary (Heap Size is 11.91 MB/992.31 MB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
57	57	2	19	57	57	6.00	0

- Cluster: 1 master + 19 slave nodes; 3 mappers, 3 reducers each
- Input: 882 compressed files each containing 100,000 sentences (88 million)
- Input stored on S3; copied to cluster at launch (3 mins)
- Actual running time: 5 hours; 67% reduction (can be increased arbitrarily)
- Total cost of running cluster = 20*6*$0.085 = $10.20!

Summary

- Python is well suited to NLP in general but missing NLP algorithms and data structures
- NLTK leverages Python and provides very powerful and convenient NLP paraphernalia
- However, NLTK doesn't scale well to "real" datasets
- Dumbo + NLTK = Best of both worlds! [Convenience + Scalability]

If you want more ...

- Replicate my experiments
  [http://www.umiacs.umd.edu/~nmadnani/pycon/replicate.pdf]
- Read Tom White's Hadoop book
  [http://oreilly.com/catalog/9780596521981]
- Read Jimmy's (upcoming) book on designing MapReduce algorithms for NLP
  [http://www.umiacs.umd.edu/~jimmylin/book.html]
- Read the lecture notes from our cloud computing course at UMD
  [http://www.umiacs.umd.edu/~jimmylin/cloud-2010-Spring/]

Questions?

http://www.qwantz.com/index.php?comic=1491