Processing Hotel Reviews with Python

& Friends

Miguel Cabrera
@mfcabrera
About Me
About Me

- Colombian
- Neuberliner
- Work for TrustYou as Data (Scientist|Engineer|Juggler)™
- Python around 2 years
- Founder and former organizer of Munich DataGeeks
Agenda
Agenda

- Problem description
- Tools
- Sample Application
TrustYou
First time here.. def won't be my last.

Trip started out with a hiccup with no limo service when I landed. But from there, this place is def the best place to stay. I had a Bellagio suite in the spa tower.. amazing views of the fountains and the strip!

Toni Santiago
4 months ago

"4 star hotel with 3 star amenities"
Reviewed yesterday NEW

Don't get me wrong this is a very luxurious hotel, however I have to say the upkeep was a tad subpar. I stayed here for 5 nights, using my MLife as a discount for the hotel. We had a very lovely Queen resort room that faced the pools. The room itself seemed fairly clean but when it came to the...
Summary of 19,720 verified reviews

88 Excellent

Excellent design hotel. Close to Las Vegas Strip. Close to restaurants, bars and shopping areas. "Great pools". "Room was clean".

Ranked TOP 10% overall

TOP 2% Design Hotel  "Perfect for design enthusiasts"  "Exclusive for conferences and meeting area"  "Beautiful lobby"

TOP 3% Luxury Hotel  "Luxury hotel"  "Free parking"  "Great view"

TOP 4% City Hotel  "Good choice for a city trip"  "Great restaurants"  "Free parking"

All languages  100%

All  100%

Couples  64%

Families  23%

Business  8%

Solo  5%

Good to know

✓ Pleasant entertainment
  "Great casino"  "Fountain show is amazing"

✓ Great pool
  "Great pools"  "Pool cafe was great"

✓ Good restaurants & bars in the area
  "Great restaurants"  "In the middle of the strip"

✓ Spacious bathroom
  "Huge bathroom"  "Bath tube was big and nice"

Review highlights

Location

"Great location"  "Shops at Caesar's Palace are just next door"  "Great restaurants"

86% enjoyed the restaurants & bars nearby.

90% liked the proximity to shopping.

Hotel

"Great hotel"  "I love the Bellagio"  "Great experience"

Ambiance

"Great casino"  "Fountain show is amazing"  "Beautiful decor"

83% were fond of the entertainment options.

89% enjoyed the friendly atmosphere.

Room

"Room was clean"  "Bellagio Suite was incredible"  "Great view"
Tasks

- Crawling
- Natural Language Processing / Semantic Analysis
- Record Linkage / Deduplication
- Ranking
- Recommendation
- Classification
- Clustering
**Batch Layer**
- Hadoop
- Python
- Pig*
- Java*

**Service Layer**
- PostgreSQL
- MongoDB
- Redis
- Cassandra

**Stack**

**Hadoop Cluster**

**Application Machines**
25 supported languages
500,000+ Properties
30,000,000+ daily crawled reviews
Deduplicated against 250,000,000+ reviews
200,000+ daily new reviews
Clean, Filter, Join and Aggregate
Crawl -> Extract
Extract -> Clean
Clean -> Stats
Stats -> ML
ML -> NLP
Steps in different technologies
Steps can be run in parallel
Steps have complex dependencies among them
Requirements

- Technology
- Parallel / Scale
- Dependency management / Orchestration
Technology
Python

- Numpy
- NLTK
- Scikit-Learn
- Pandas
- IPython / Jupyter
Scaling
Hadoop = Java?
Python + Hadoop

- Hadoop Streaming
- MRJob
- Oozie
- Luigi
- ...

...
Hadoop Streaming

cat input.txt | ./map.py | sort | ./reduce.py > output.txt
Hadoop Streaming

hadoop jar contrib/streaming/hadoop-*/streaming*.jar \\n-file /home/hduser/mapper.py -mapper /home/hduser/mapper.py \\n-file /home/hduser/reducer.py -reducer /home/hduser/reducer.py \\n-input /user/hduser/text.txt -output /user/hduser/gutenberg-output
Who likes to write Bash scripts?
Orchestrate
Luigi

“A python framework for data flow definition and execution”
Luigi

- Dependency definition
- Hadoop / HDFS Integration
- Object oriented abstraction
- Parallelism
- Resume failed jobs
- Visualization of pipelines
- Command line integration
```python
class WordCount(luigi.Task):
    date = luigi.DateParameter()

def requires(self):
    return InputText(date)

def output(self):
    return luigi.LocalTarget('/tmp/%s' % self.date_interval)

def run(self):
    count = {}

    for f in self.input():
        for line in f.open('r'):
            for word in line.strip().split():
                count[word] = count.get(word, 0) + 1

        f = self.output().open('w')
    for word, count in six.iteritems(count):
        f.write("%s\t%d\n" % (word, count))
    f.close()
```
```python
class WordCount(luigi.Task):
    date = luigi.DateParameter()

def requires(self):
    return InputText(date)

def output(self):
    return luigi.LocalTarget('/tmp/%s' % self.date_interval)

def run(self):
    count = {}

    for f in self.input():
        for line in f.open('r').split():
            for word in line.strip().split():
                count[word] = count.get(word, 0) + 1

    f = self.output().open('w')
    for word, count in six.iteritems(count):
        f.write('%s%td\n' % (word, count))
    f.close()
```

**Task Parameters**
class WordCount(luigi.Task):
    date = luigi.DateParameter()

    def requires(self):
        return InputText(date)

    def output(self):
        return luigi.LocalTarget('/tmp/%s' % self.date_interval)

    def run(self):
        count = {}

        for f in self.input():
            for line in f.open('r'):
                for word in line.strip().split():
                    count[word] = count.get(word, 0) + 1

        f = self.output().open('w')
        for word, count in six.iteritems(count):
            f.write('%s\t%d
' % (word, count))
        f.close()
class WordCount(luigi.Task):
    date = luigi.DateParameter()

def requires(self):
    return InputText(date)

def output(self):
    return luigi.LocalTarget('/tmp/%s' % self.date_interval)

def run(self):
    count = {}

    for f in self.input():
        for line in f.open('r').split():
            count[word] = count.get(word, 0) + 1

    f = self.output().open('w')
    for word, count in six.iteritems(count):
        f.write("%s\t%d\n" % (word, count))
    f.close()
class WordCount(luigi.Task):
    date = luigi.DateParameter()

    def requires(self):
        return InputText(date)

    def output(self):
        return luigi.LocalTarget('/tmp/%s' % self.date_interval)

    def run(self):
        count = {}

        for f in self.input():
            for line in f.open('r').split():
                for word in line.strip().split():
                    count[word] = count.get(word, 0) + 1

        f = self.output().open('w')
        for word, count in six.iteritems(count):
            f.write('%s\td
' % (word, count))
        f.close()
class WordCount(luigi.hadoop.JobTask):
    date = luigi.DateParameter()

    def requires(self):
        return InputText(date)

    def output(self):
        return luigi.hdfs.HdfsTarget('%s' % self.date_interval)

    def mapper(self, line):
        for word in line.strip().split():
            yield word, 1

    def reducer(self, key, values):
        yield key, sum(values)
Luigi

- Minimal boilerplate code
- Programmatically define dependencies
- Integration with HDFS / Hadoop
- Task Synchronization
- Can wrap anything
Before

- **Bash scripts + Cron**
- **Manual cleanup**
- **Manual failure recovery**
- **Hard(er) to debug**
Now

- Complex nested Luigi jobs graphs
- Automatic retries
- Still Hard to debug
We use it for...

- Standalone executables
- Dump data from databases
- General Hadoop Streaming
- Bash Scripts / MRJob
- Pig* Scripts
You can wrap anything
You can wrap anything
The right tool for the right job
Pig is a highlevel platform for creating MapReduce programs with Hadoop
SQL

SELECT f3, SUM(f2), AVG(f1) FROM relation WHERE f1 > 500 GROUP BY f3

Pig Latin

rel = LOAD 'relation' AS (f1: int, f2: int, f3: chararray);
rel = FILTER rel f1 > 500
by_f3 = GROUP rel BY f3;
result = FOREACH by_f3 GENERATE group, SUM(by_f3.f2), AVG(by_f3.f1)

Python

def map(r):
    if r['f1'] > 500:
        yield r['f3'], [r['f1'], r['f2']]  
def reduce(k, values):
    avg = 0
    summ = 0
    l = len(values)
    for r in values:
        summ += r[1]
        avg += r[0]
    avg = avg/float(l)
    yield k, [summ, avg]
Pig + Python

- Data loading and transformation in Pig
- Other logic in Python
- Pig as a Luigi Task
- Pig UDFs defined in Python
Sample Application
Reviews are boring...
The Mountain Three Wolf Moon Short Sleeve Tee

Price: $8.15 - $32.95 & FREE Returns on some sizes and colors. Details
Sale: Lower price available on select options

Size:
Select ▼ Size Chart | Fit: As expected (91%) ▼

Color: Blue

- Cotton
- Machine Wash
- 100% Cotton
- Exceptional artwork on a tee shirt
- Comfortable, and durable
- Machine wash cold, tumble dry low, do not bleach
- Use/Mexico

Show more
Thank You......

I was a normal guy with a normal life before I purchased this shirt. The day it came in the mail and I looked in the mirror at myself in it, I looked so good in this shirt that I... Read more

39,048 of 39,561 people found the following review helpful

Dual Function Design

By Amazon Customer on November 10, 2008

Size: Boys 2-4  |  Color: Dark Green

This item has wolves on it which makes it intrinsically sweet and worth 5 stars by itself, but once I tried it on, that's when the magic happened. After checking to ensure that the shirt would properly cover my girth, I walked from my trailer to Walmart with the shirt on and was immediately approached by women. The women knew from the wolves on my shirt that I, like a wolf, am a mysterious loner who knows how to 'howl at the moon' from time to time (if you catch my drift!). The women that approached me wanted to know if I would be their boyfriend and/or give them money for something they called meth. I told them no, because they didn't have enough teeth, and frankly a man with a wolf-shirt shouldn't settle for the first thing that comes to him.

978 of 1,078 people found the following review helpful

With Great Powers Comes Great Responsibility

By Amazon Customer on July 17, 2009

I admit it, I'm a ladies' man. And when you put this shirt on a ladies' man, it's like giving an AK-47 to a ninja. Sure it looks cool and probably would make for a good movie, but you know somebody is probably going to get hurt in the end (no pun intended). That's what almost happened to me, this is my story...
“There was poo in the kettle...”
Reviewed 11 September 2011 via mobile

Do not stay in this absolute dump of a hotel, and I use the term hotel very loosely. I cannot convey strongly enough how disgusting this place is. Blood stained headboards that have clearly been up since the world war (the first one), rude staff, windows that won't close, no hot water, broken furniture, dirty utensils, broken light fixings and actual poo in the kettle. Pretty sure I'm going to end up with some sort of rash/disease due to sanitation conditions similar to those of a homeless crack head. In summary...this place is a complete hole.

“They spat in my hair”
Reviewed 28 February 2013

“haunted”
Reviewed 1 February 2012 via mobile

My girlfriend were having a romantic meal, however it was spoilt when an employee spat in my hair. CHEEKY OR WHAT?

stay out of it its haunted i saw the ghost in my room over my 4 yr boy i screamed and ran out my room we collected our thing and checked out in the middle of the night the rude man in the front desk wanted to charge us for a day while we only stayed 6 hrs after checking out my sister told me she heared water flushing in the toilet and my nanny saw the tv channels switching by it self while my kids watching tom and jerry but they didnt think that it really haunted our room# was 303
Reviews highlight the individuality and personality of users.
Snippets from Reviews

“Hips don’t lie”
“Maid was banging”
“Beautiful bowl flowers”
“Irish dance, I love that”
“No ghost sighting”
“One ghost touching”
“Too much cardio, not enough squats in the gym”
“it is like hugging a bony super model”
Word2Vec
Group of algorithms
An instance of shallow learning
Feature learning model
Generates real-valued vectors representation of words
“king” − “man” + “woman” = “queen”
Word2Vec

Load up the word vectors

QUEEN \([0.3, 0.9]\)

KING \([0.5, 0.7]\)

WOMAN \([0.3, 0.4]\)

MAN \([0.5, 0.2]\)

Source: http://technology.stitchfix.com/blog/2015/03/11/word-is-worth-a-thousand-vectors/
Word2Vec

Start with man - woman

KING

WOMAN [0.3, 0.4]

MAN [0.5, 0.2]

Source: http://technology.stitchfix.com/blog/2015/03/11/word-is-worth-a-thousand-vectors/
Word2Vec

Start with \text{man} - \text{woman}

Source: http://technology.stitchfix.com/blog/2015/03/11/word-is-worth-a-thousand-vectors/
Word2Vec

Then take king

KING [0.5, 0.7]

MAN - WOMAN

Source: http://technology.stitchfix.com/blog/2015/03/11/word-is-worth-a-thousand-vectors/
Word2Vec

And add \text{man - woman}

\text{? \ [0.3, 0.9]} \quad \text{MAN - WOMAN \ [0.5, 0.7]}
Word2Vec

queen is closest to resulting vector

QUEEN [0.3, 0.9]

KING

MAN - WOMAN

Source: http://technology.stitchfix.com/blog/2015/03/11/word-is-worth-a-thousand-vectors/
Similar words are nearby vectors
Wor2vec offer a similarity metric of words
Can be extended to paragraphs and documents
A fast Python based implementation available via Gensim
Hotel Reviews + Gensim + Python + Luigi = ?
from gensim.models.doc2vec import Doc2Vec

class LearnModelTask(luigi.Task):
    # Parameters.... blah blah blah

    def output(self):
        return luigi.LocalTarget(os.path.join(self.output_directory, self.model_out))

    def requires(self):
        return LearnBigramsTask()

    def run(self):
        sentences = LabeledClusterIDSentence(self.input().path)
        model = Doc2Vec(sentences=sentences,
                        size=int(self.size),
                        dm=int(self.distmem),
                        negative=int(self.negative),
                        workers=int(self.workers),
                        window=int(self.window),
                        min_count=int(self.min_count),
                        train_words=True)
        model.save(self.output().path)
Wor2vec/Doc2vec offer a similarity metric of words
Similarities are useful for non-personalized recommender systems
Non-personalized recommenders recommend items based on what other consumers have said about the items.
http://demo.trustyou.com
Takeaways
Takeaways

• It is possible to use Python as the primary language for doing large data processing on Hadoop.
• It is not a perfect setup but works well most of the time.
• Keep your ecosystem open to other technologies.
• Products reviews contain much more information than just facts.
Questions?