# Apache Pig for Data Science

**Casey Stella**

Hortonworks

June, 2014

# Table of Contents

# Introduction

- I'm a Principal Architect at Hortonworks
- I work primarily doing Data Science in the Hadoop Ecosystem
- Prior to this, I've spent my time and had a lot of fun
  - Doing data mining on medical data at Explorys using the Hadoop ecosystem
  - Doing signal processing on seismic data at Ion Geophysical using MapReduce
  - Being a graduate student in the Math department at Texas A&M in algorithmic complexity theory
- I'm going to talk about Apache Pig's role for doing scalable data science.

# Apache Pig: What is it?

Pig is a high level scripting language for operating on large datasets inside Hadoop

## Apache Pig: What is it?

Pig is a high level scripting language for operating on large datasets inside Hadoop

- Transforms high level data operations into MapReduce/Tez jobs

# Apache Pig: What is it?

Pig is a high level scripting language for operating on large datasets inside Hadoop

- Transforms high level data operations into MapReduce/Tez jobs
- Optimizes such that the minimal number of MapReduce/Tez jobs need be run

# Apache Pig: What is it?

Pig is a high level scripting language for operating on large datasets inside Hadoop

- Transforms high level data operations into MapReduce/Tez jobs
- Optimizes such that the minimal number of MapReduce/Tez jobs need be run
- Familiar relational primitives available

# Apache Pig: What is it?

Pig is a high level scripting language for operating on large datasets inside Hadoop

- Transforms high level data operations into MapReduce/Tez jobs
- Optimizes such that the minimal number of MapReduce/Tez jobs need be run
- Familiar relational primitives available
- Extensible via User Defined Functions and Loaders for customized data processing and formats

# Apache Pig: An Familiar Example

```
SENTENCES= load '...' as (sentence:chararray);
WORDS = foreach SENTENCES
        generate flatten(TOKENIZE(sentence))
        as word;
WORD_GROUPS = group WORDS by word;
WORD_COUNTS = foreach WORD_GROUPS
              generate group as word, COUNT(WORDS);
store WORD_COUNTS into '...';
```

## Understanding Data

"80% of the work in any data project is in cleaning the data."

— D.J. Patel in *Data Jujitsu*

# Understanding Data

A core pre-requisite to analyzing data is understanding data's shape and distribution. This requires (among other things):

- Computing distribution statistics on data
- Sampling data

# Understanding Data: Datafu

An Apache Incubating project called **datafu**[1] provides some of these tooling in the form of Pig UDFs:

- Computing quantiles of data

[1] http://datafu.incubator.apache.org/

# Understanding Data: Datafu

An Apache Incubating project called **datafu**[1] provides some of these tooling in the form of Pig UDFs:

- Computing quantiles of data
- Sampling
  - Bernoulli sampling by probability (built into pig)

[1] http://datafu.incubator.apache.org/

# Understanding Data: Datafu

An Apache Incubating project called **datafu**[1] provides some of these tooling in the form of Pig UDFs:

- Computing quantiles of data
- Sampling
  - Bernoulli sampling by probability (built into pig)
  - Simple Random Sample

[1] http://datafu.incubator.apache.org/

# Understanding Data: Datafu

An Apache Incubating project called **datafu**[1] provides some of these tooling in the form of Pig UDFs:

- Computing quantiles of data
- Sampling
  - Bernoulli sampling by probability (built into pig)
  - Simple Random Sample
  - Reservoir sampling

[1] http://datafu.incubator.apache.org/

# Understanding Data: Datafu

An Apache Incubating project called **datafu**[1] provides some of these tooling in the form of Pig UDFs:

- Computing quantiles of data
- Sampling
  - Bernoulli sampling by probability (built into pig)
  - Simple Random Sample
  - Reservoir sampling
  - Weighted sampling without replacement

[1] http://datafu.incubator.apache.org/

## Understanding Data: Datafu

An Apache Incubating project called **datafu**[1] provides some of these tooling in the form of Pig UDFs:

- Computing quantiles of data
- Sampling
  - Bernoulli sampling by probability (built into pig)
  - Simple Random Sample
  - Reservoir sampling
  - Weighted sampling without replacement
  - Random Sample with replacement

[1] http://datafu.incubator.apache.org/

# Case Study: Bootstrapping

**Bootstrapping** is a resampling technique which is intended to measure accuracy of sample estimates. It does this by measuring an estimator (such as mean) across a set of random samples with replacement from an original (possibly large) dataset.

# Case Study: Bootstrapping

Datafu provides two tools which can be used together to provide that random sample with replacement:

- SimpleRandomSampleWithReplacementVote – Ranks multiple candidates for each position in a sample
- SimpleRandomSampleWithReplacementElect – Chooses, for each position in the sample, the candidate with the lowest score

The datafu docs provide an example[2] of generating a boostrap of the mean estimator.

[2] http://datafu.incubator.apache.org/docs/datafu/guide/sampling.html

# What is Machine Learning?

Machine learning is the study of systems that can learn from data.
The general tasks fall into one of two categories:

# What is Machine Learning?

Machine learning is the study of systems that can learn from data. The general tasks fall into one of two categories:

- Unsupervised Learning
    - Clustering
    - Outlier detection
    - Market Basket Analysis

# What is Machine Learning?

Machine learning is the study of systems that can learn from data. The general tasks fall into one of two categories:

- Unsupervised Learning
  - Clustering
  - Outlier detection
  - Market Basket Analysis
- Supervised Learning
  - Classification
  - Regression
  - Recommendation

# Building Machine Learning Models with Pig

Machine Learning at scale in Hadoop generally falls into two categories:

- Build one large model on all (or almost all) of the data
- Sample the large dataset and build the model based on that sample

# Building Machine Learning Models with Pig

Machine Learning at scale in Hadoop generally falls into two categories:

- Build one large model on all (or almost all) of the data
- Sample the large dataset and build the model based on that sample

Pig can assist in intelligently sampling down the large data into a training set. You can then use your favorite ML algorithm (which can be run on the JVM) to generate a machine learning model.

## Applying Models with Pig

Pig shines at batch application of an existing ML model. This generally is of the form:

- Train a model out-of-band

# Applying Models with Pig

Pig shines at batch application of an existing ML model. This generally is of the form:

- Train a model out-of-band
- Write a UDF in Java or another JVM language which can apply the model to a data point

# Applying Models with Pig

Pig shines at batch application of an existing ML model. This generally is of the form:

- Train a model out-of-band
- Write a UDF in Java or another JVM language which can apply the model to a data point
- Call the UDF from a pig script to distribute the application of the model across your data in parallel

# What is Natural Language Processing?

- Natural language processing is the field of Computer Science, Linguistics & Math that covers computer understanding and manipulation of human language.
  - Historically, linguists hand-coded rules to accomplish much analysis
  - Most modern approaches involves using Machine Learning

# What is Natural Language Processing?

- Natural language processing is the field of Computer Science, Linguistics & Math that covers computer understanding and manipulation of human language.
  - Historically, linguists hand-coded rules to accomplish much analysis
  - Most modern approaches involves using Machine Learning

- Mature field with many useful libraries on the JVM
  - Apache OpenNLP
  - Stanford CoreNLP
  - MALLET

# Natural Language Processing with Large Data

- Generally low-volume, complex analysis
  - Big companies often don't have a ton of natural language data
  - Dropped previously because they were unable to analyze

# Natural Language Processing with Large Data

- Generally low-volume, complex analysis
  - Big companies often don't have a ton of natural language data
  - Dropped previously because they were unable to analyze

- Sometimes high-volume, complex analysis
  - Search Engines
  - Social media content analysis

# Natural Language Processing with Large Data

- Generally low-volume, complex analysis
  - Big companies often don't have a ton of natural language data
  - Dropped previously because they were unable to analyze
- Sometimes high-volume, complex analysis
  - Search Engines
  - Social media content analysis
- Typically many small-data problems in parallel
  - Often requires only the context of a single document
  - Ideal for encapsulating as Pig UDFs

## Natural Language Processing: Demo

- Stanford CoreNLP integrated the work of Richard Socher, et al [2] using recursive deep neural networks to predict sentiment of movie reviews.
- There is a large set of IMDB movie reviews used to analyze sentiment analysis [1].
- Let's look at how to encapsulate this into a Pig UDF and run on some movie review data.

# The UDF

```
public class ANALYZE_SENTIMENT extends EvalFunc<String>
{
    @Override
    public String exec(Tuple objects) throws IOException
    {
        String document = (String)objects.get(0);
        if(document == null || document.length() == 0)
        {
            return null;
        }
        //Call out to our handler that we wrote to do the
            sentiment analysis
        SentimentClass sentimentClass = SentimentAnalyzer.INSTANCE
            .apply(document);
        return sentimentClass == null?null: sentimentClass.
            toString();
    }
}
```

# The Pig Script

```
REGISTER ./Pig_for_Data_Science-1.0-SNAPSHOT.jar
DEFINE SENTIMENT_ANALYSIS com.caseystella.ds.pig.ANALYZE_SENTIMENT
    ;

DOCUMENTS_POS = LOAD ...
DOCUMENTS_NEG = LOAD ...

NEG_DOCS_WITH_SENTIMENT = foreach DOCUMENTS_NEG generate ’NEGATIVE
    ’ as true_sentiment
      , document as document;
POS_DOCS_WITH_SENTIMENT = foreach DOCUMENTS_POS generate ’POSITIVE
    ’ as true_sentiment
      , document as document;
DOCS_WITH_SENTIMENT = UNION NEG_DOCS_WITH_SENTIMENT,
    POS_DOCS_WITH_SENTIMENT;

PREDICTED_SENTIMENT = foreach DOCS_WITH_SENTIMENT generate
    SENTIMENT_ANALYSIS(document) as predicted_sentiment
                                        , true_sentiment
                                       , document;
STORE PREDICTED_SENTIMENT INTO ...
```

# Results

- Executing on a sample of size 1022 Positive and Negative documents.
- Overall Accuracy of 77.2%

| | | Actual | | |
|---|---|---|---|---|
| | | Positive | Negative | Total |
| Predicted | Positive | 367 | 114 | 481 |
| | Negative | 119 | 422 | 541 |
| | Total | 486 | 536 | 1022 |

# Questions

Thanks for your attention! Questions?

- Code & scripts for this talk available on my github presentation page.[3]
- Find me at http://caseystella.com
- Twitter handle: @casey_stella
- Email address: cstella@hortonworks.com

[3] http://github.com/cestella/presentations/

# Bibliography

[1] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[2] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA, October 2013. Association for Computational Linguistics.