SPARK+AI SUMMIT 2018

Spark NLP: Extending Spark ML to Deliver Fast, Scalable & Unified Natural Language Processing

Alex Thomas
Data Scientist @ Indeed

David Talby
CTO @ Pacific AI

#DS1SAIS

CONTENTS

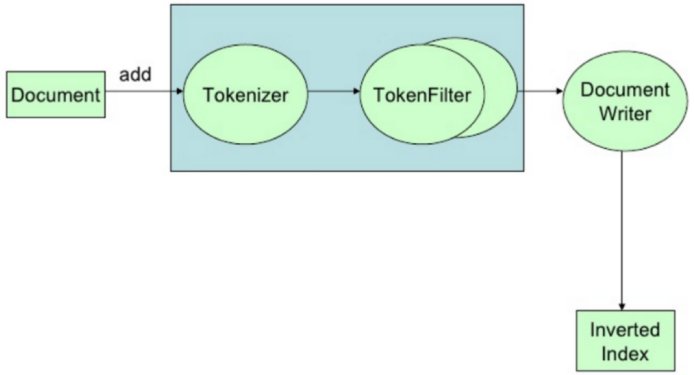
- ✓ WHAT IS NATURAL LANGUAGE UNDERSTANDING?
- ✓ INTRODUCING SPARK NLP
- ✓ CODE WALKTHROUGH: SENTIMENT ANALYSIS

1.

What is Natural Language Understanding?

AT THE BEGINNING, THERE WAS SEARCH

- Query examples:
  - jazoon
  - jazoon AND java <=> +jazoon +java
  - jazoon OR java
  - jazoon NOT php <=> jazoon -php
  - conference AND (java OR j2ee)
  - "Java conference"
  - title:jazoon
  - j?zoon
  - jaz*
  - schmidt~ schmidt, schmit, schmitt
  - price:[000 TO 050]



Scalable & robust Indexing pipeline
Tokenizers & analyzers
Synonyms, spellers & Auto-suggest
File formats & header boosting
Rankers, link & reputation boosting

THEN, THERE WAS SEMANTIC SEARCH

```
“cheap red prom dresses”
“laptops under $500”
“italian restaurants that deliver here”
“black panther tonight”
“nba scores”
```

THEN, YOU NEED TO UNDERSTAND LANGUAGE

`Prescribing sick days due to diagnosis of influenza.`	*Positive*
`Jane complains about flu-like symptoms.`	*Speculative*
`Jane’s RIDT came back clean.`	*Negative*
`Jane is at risk for flu if she’s not vaccinated.`	*Conditional*
`Jane’s older brother had the flu last month.`	*Family history*
`Jane had a severe case of flu last year.`	*Patient history*

Parts of Speech • Dependency Parsing • Coreference Resolution • Entity Recognition

WHAT MAKES LANGUAGE HARD

- **Nuanced**
  - Sure / I agree / Absolutely! / Whatever / Yes sir / Just to see you smile ❤️
- **Fuzzy**
  - Blue, New, Tall, Child, Tell, Do
- **Contextual**
  - “Patient denies alcohol abuse”
- **Medium specific**
  - “SGTM c u in 15”
- **Domain specific**
  - *All forward-looking statements included in this document are based on information available to us on the date hereof, and we assume no obligation to revise or publicly release any revision to any such forward-looking statement, except as may otherwise be required by law.*

USE CASES

	Get by with rules, search, RegEx, attribute extraction	Welcome to the world of NLP, ML and DL
Social media	Does this social media post contain an offensive word?	Is this social media post offensive?
Legal	Find patents with the terms ‘car’ and ‘battery’, or synonyms	Who is patenting next-gen electrical car batteries?
Support	Find products mentioned in customer emails or phone calls	What is this customer complaining about?
Finance	Extract the fee structure from a mutual fund prospectus	Are UK pensions allowed to invest in this fund?
Healthcare	Extract the patient’s blood pressure reading from a note	Does this patient have high blood pressure?

2.

Introducing Spark NLP

INTRODUCING SPARK NLP

- Industrial Grade NLP for the Spark ecosystem
- Design Goals:
  1. **Performance & Scale**
  2. **Frictionless Reuse**
  3. **Enterprise Grade**
- Built on top of the Spark ML API's
- Apache 2.0 licensed, with active development & support

NATIVE SPARK EXTENSION

High Performance Natural Language Understanding at Scale

John Snow LABS

Part of Speech Tagger
Named Entity Recognition
Sentiment Analysis
Spell Checker
Tokenizer
Stemmer
Lemmatizer
Entity Extraction

Spark MLlib

Topic Modeling
Word2Vec
TF-IDF
String distance calculation
N-grams calculation
Stop word removal
Train/Test & Cross-Validate
Ensembles

Spark ML API (Pipeline, Transformer, Estimator)

Spark SQL API (DataFrame, Catalyst Optimizer)

Spark Core API (RDD’s, Project Tungsten)

Data Sources API

FRICTIONLESS REUSE

```
pipeline = pyspark.ml.Pipeline(stages=[
                document_assembler,
                tokenizer,
                stemmer,
                normalizer,
                stopword_remover,
                tf,
                idf,
                lda])

topic_model = pipeline.fit(df)
```

Spark NLP annotators

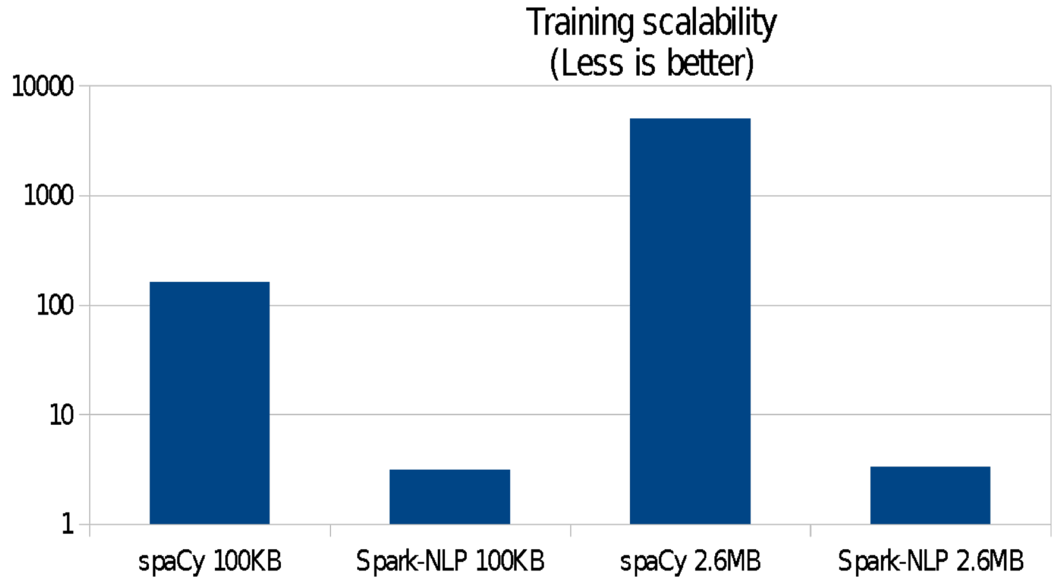
Spark ML featurizers

Spark ML LDA implementation

Single execution plan for the given data frame

BENCHMARK: TRAINING

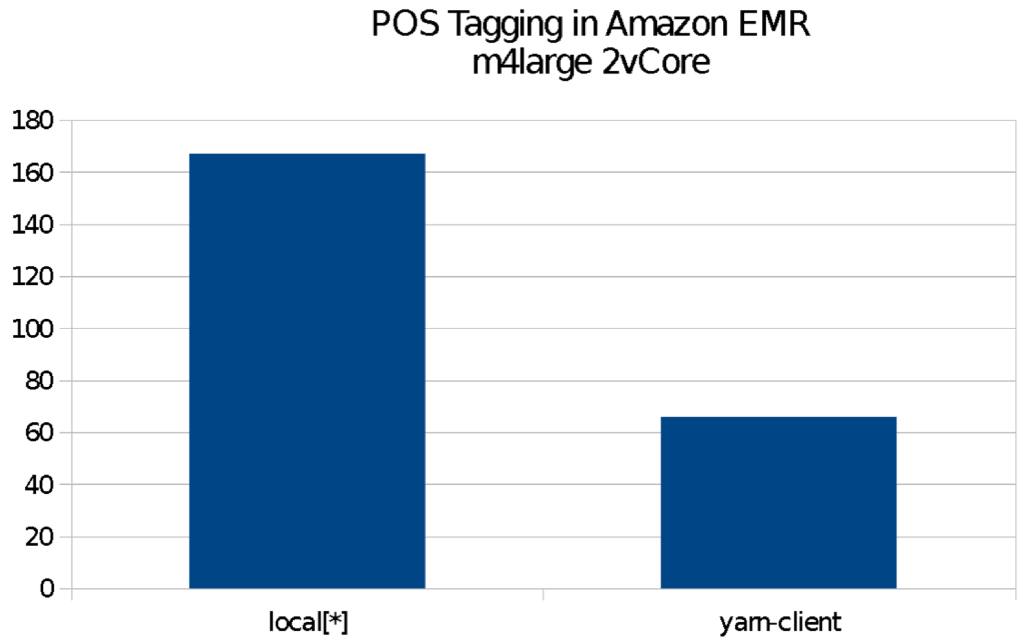
- Run on a desktop PC, Linux Mint with 16GB RAM, local SSD drives, & Intel core i5-6600K processor running 4 cores at 3.5GHz
- Data has been taken from the National American Corpus (http://www.anc.org), utilizing the MASC 3.0.2 written corpora from the newspaper section.
- Pipeline has Sentence Boundary, Tokenization & Part of Speech



- **Spark-NLP was 38 times faster to train on 100kb of data**
- **Spark-NLP was 80 times faster to train on 2.6mb of data**

BENCHMARK: SCALING

- Spark-NLP against itself
- 2.5x speedup with a 4-node cluster
- Zero code changes
- Spark scales as Spark does:
  1 to 3 orders of magnitude faster
  depending on cluster setup
- Not compares to spaCy, since it cannot leverage a cluster



SPARK NLP 1.5 IMPROVEMENTS

Light Pipelines

10x speedup for 'small data' (<= 50k documents)

Scalable Pipelines

Only open source cluster distributed NLP (>= 5M documents)

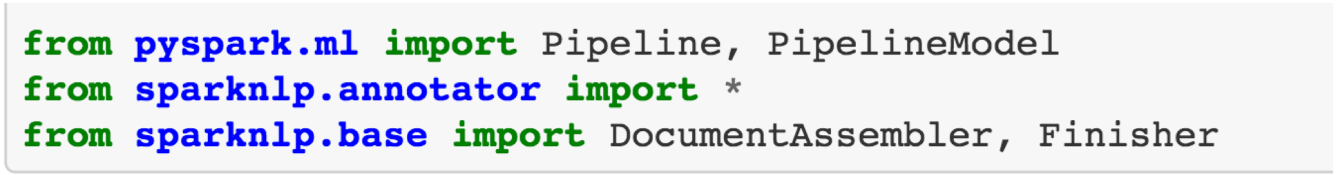
3.

Code Walkthrough: Sentiment Analysis

USING SPARK NLP

- Homepage: https://nlp.johnsnowlabs.com
  - Getting Started, Documentation, Examples, Videos, Blogs
  - Join the Slack Community
- GitHub: https://github.com/johnsnowlabs/spark-nlp
  - Open Issues & Feature Requests
  - Contribute!
- The library has Scala and Python 2 & 3 API's
- Get directly from maven-central or spark-packages
- Tested on all Spark 2.x versions

SENTIMENT ANALYSIS IN PYTHON: SETTING UP



```python
from pyspark.ml import Pipeline, PipelineModel
from sparknlp.annotator import *
from sparknlp.base import DocumentAssembler, Finisher
```

CONFIGURING THE ANNOTATIONS PIPELINE

We creat the document assemblerr, which will put target text column into Annotation form

```
### Define the dataframe
document_assembler = DocumentAssembler() \
            .setInputCol("text")

### Transform input to appropriate schema
#assembled = document_assembler.transform(data)
```

The sentence detector will parse sub sentences in every line

```
### Sentence detector
sentence_detector = SentenceDetector() \
    .setInputCols(["document"]) \
    .setOutputCol("sentence")
#sentence_data = sentence_detector.transform(checked)
```

CONFIGURING THE ANNOTATIONS PIPELINE

The tokenizer will match standard tokens

```
### Tokenizer
tokenizer = Tokenizer() \
            .setInputCols(["sentence"]) \
            .setOutputCol("token")
#tokenized = tokenizer.transform(assembled)
```

Normalizer will clean out the tokens

```
normalizer = Normalizer() \
             .setInputCols(["token"]) \
             .setOutputCol("normal")
```

CONFIGURING THE ANNOTATIONS PIPELINE

The spell checker will correct normalized tokens, this trains with a dictionary of english words

```
### Spell Checker
spell_checker = NorvigSweetingApproach() \
            .setInputCols(["normal"]) \
            .setOutputCol("spell") \
            .setDictionary("file:///" + os.getcwd() + "/../../../src/test/resources/spell/words.tx
t")
```

CONFIGURING THE ANNOTATIONS PIPELINE

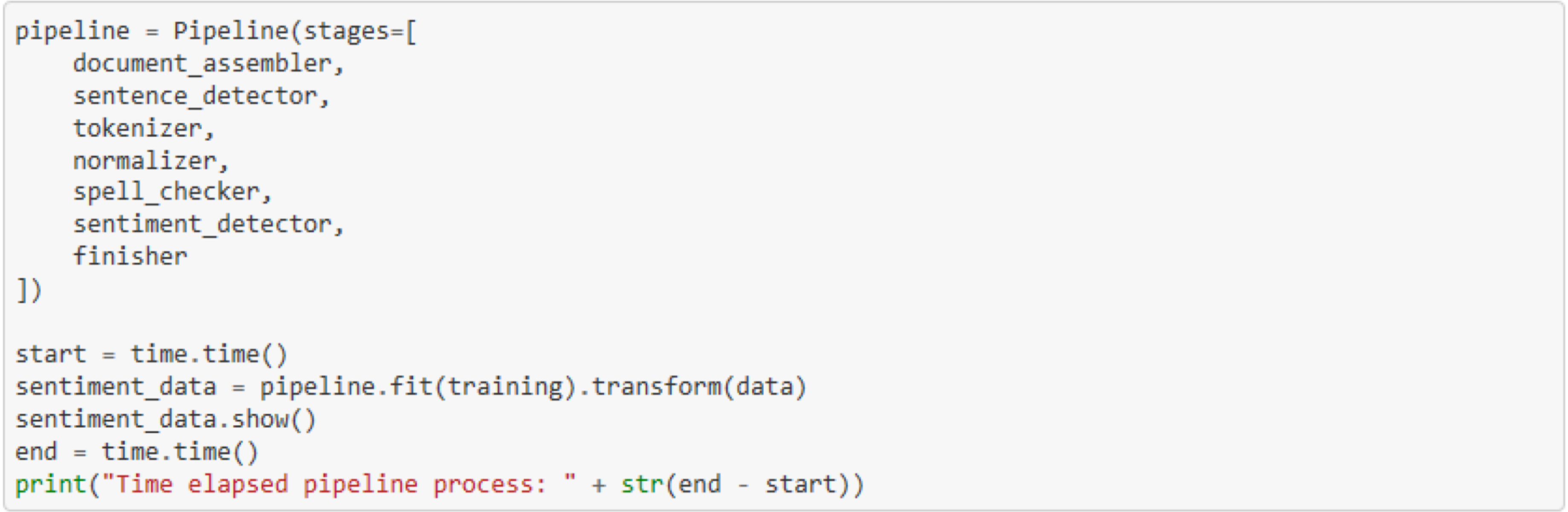
We creat the ViveknSentimentApproach and set resources to train it

```
sentiment_detector = ViveknSentimentApproach() \
    .setInputCols(["spell", "sentence"]) \
    .setOutputCol("sentiment") \
    .setPruneCorpus(0) \
    .setPositiveSource("file:///" + os.getcwd() + "/../../../src/test/resources/vivekn/positive")
 \
    .setNegativeSource("file:///" + os.getcwd() + "/../../../src/test/resources/vivekn/negative")
 \
```

The finisher will utilize sentiment analysis output

```
finisher = Finisher() \
    .setInputCols(["sentiment"]) \
    .setIncludeKeys(True)
```

TRAINING & RUNNING THE PIPELINE



```
pipeline = Pipeline(stages=[
    document_assembler,
    sentence_detector,
    tokenizer,
    normalizer,
    spell_checker,
    sentiment_detector,
    finisher
])

start = time.time()
sentiment_data = pipeline.fit(training).transform(data)
sentiment_data.show()
end = time.time()
print("Time elapsed pipeline process: " + str(end - start))
```

LEARN MORE: NLP DEEP DIVE IS AT 2PM TODAY

"STATE OF THE ART NATURAL LANGUAGE PROCESSING AT SCALE"

- Document classification example: Unified NLP & ML pipeline
- Word vectors & feature engineering from text
- Healthcare-specific NLP annotators
- Training your own deep learning NLP models

THANK YOU!

althomas@indeed.com

in/alnith/

david@pacific.ai

in/davidtalby

@davidtalby