RELATIONSHIP EXTRACTION FROM UNSTRUCTURED TEXT-BASED ON STANFORD NLP WITH SPARK

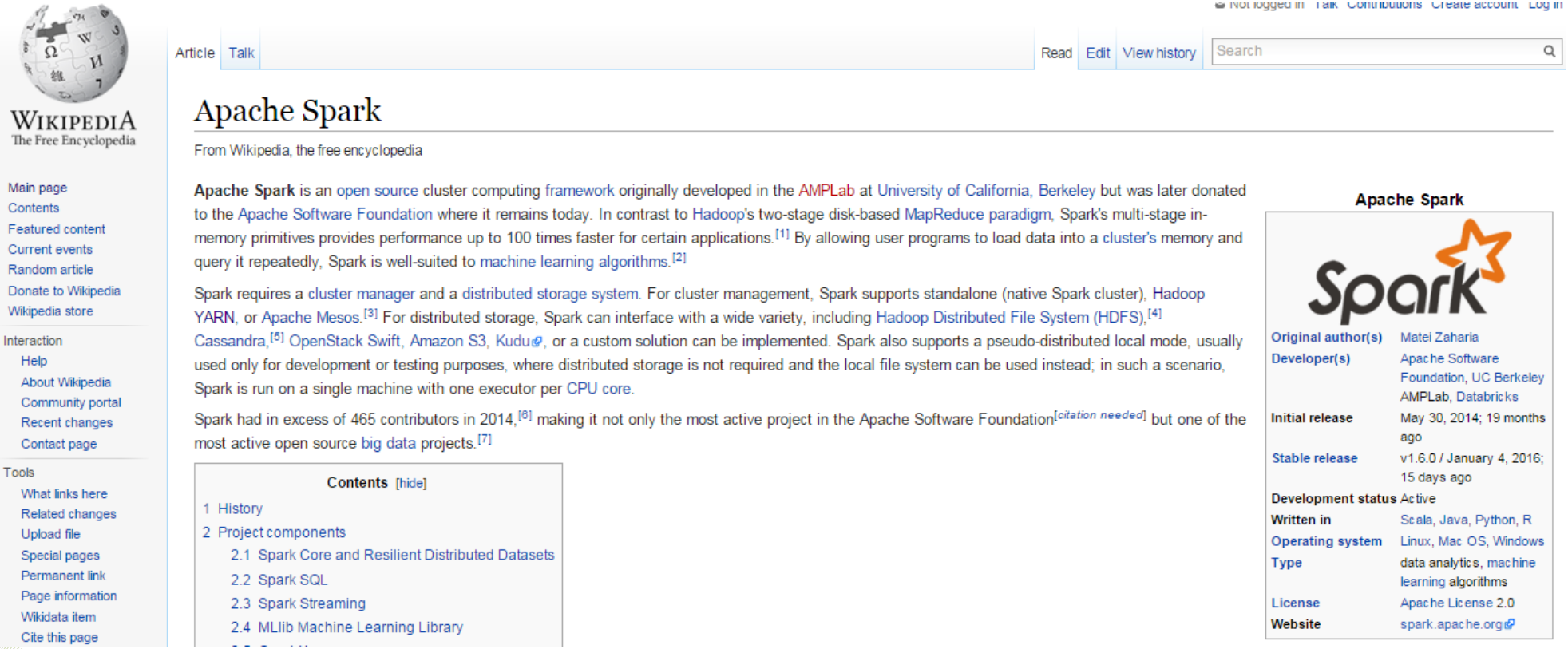
Yana Ponomarova
Head of Data Science France - Capgemini
Nicolas Claudon
Head of Big Data Architects France - Capgemini

SPARK SUMMIT EAST
DATA SCIENCE AND ENGINEERING AT SCALE
FEBRUARY 16-18, 2016 NEW YORK CITY

Taming the Text

- 80% of world's information is in a form of text
- Text documents are highly unstructured
- Search engines do not satisfy all information extraction needs

What if you could « structure » text....



WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Not logged in Talk Contributions Create account Log in

Article Talk | Read Edit View history | Search

Apache Spark

From Wikipedia, the free encyclopedia

Apache Spark is an open source cluster computing framework originally developed in the AMPLab at University of California, Berkeley but was later donated to the Apache Software Foundation where it remains today. In contrast to Hadoop's two-stage disk-based MapReduce paradigm, Spark's multi-stage in-memory primitives provides performance up to 100 times faster for certain applications.[1] By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well-suited to machine learning algorithms.[2]

Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports standalone (native Spark cluster), Hadoop YARN, or Apache Mesos.[3] For distributed storage, Spark can interface with a wide variety, including Hadoop Distributed File System (HDFS),[4] Cassandra,[5] OpenStack Swift, Amazon S3, Kudu, or a custom solution can be implemented. Spark also supports a pseudo-distributed local mode, usually used only for development or testing purposes, where distributed storage is not required and the local file system can be used instead; in such a scenario, Spark is run on a single machine with one executor per CPU core.

Spark had in excess of 465 contributors in 2014,[6] making it not only the most active project in the Apache Software Foundation[*citation needed*] but one of the most active open source big data projects.[7]

Contents [hide]
1 History
2 Project components
2.1 Spark Core and Resilient Distributed Datasets
2.2 Spark SQL
2.3 Spark Streaming
2.4 MLlib Machine Learning Library

Apache Spark

Original author(s)	Matei Zaharia
Developer(s)	Apache Software Foundation, UC Berkeley AMPLab, Databricks
Initial release	May 30, 2014; 19 months ago
Stable release	v1.6.0 / January 4, 2016; 15 days ago
Development status	Active
Written in	Scala, Java, Python, R
Operating system	Linux, Mac OS, Windows
Type	data analytics, machine learning algorithms
License	Apache License 2.0
Website	spark.apache.org

Use Cases and Benefits

- Querying Supply Chain graph

  *Select all **receivers** of **crude oil** from **site A***

- Alerts for undesirable contract commitements

  « ***The Defence Ministry*** *has decided to impose* ***unlimited penalty*** *on* ***foreign vendors***»

- Alerts for desirable opportunities from your financial news feeds

  « ***Macquarie*** *upgraded* ***AUY*** *from* ***Neutral*** *to* ***Outperform*** *rating* »

Relation Extraction : Approaches

- ACE : 17 rel-s (role, contain, location, family)
- UMLS : 134 entity types, 54 relations
- Freebase : nationality, contains, profession, place of birth, etc.
- Approaches (D. Jurafsky : *Speech and Language Processing*)

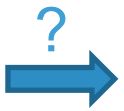
1.	Search for patterns	⇨	***Our approach***
2.	Supervised machine learning	⇨	Need training set
3.	Semi-supervised and unsupervised	⇨	Need training set (smaller)
4.	Deep Learning	⇨	Need huge training sets

Relation Extraction : Search for Pattern

- Hearst Patterns : Ontology construction

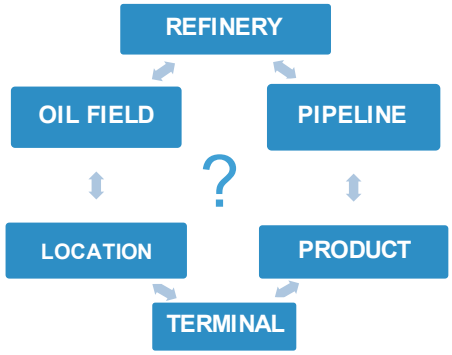
X and other Y	...temples, treasuries, and other important civic buildings.
X or other Y	Bruises, wounds, broken bones or other injuries...
Y such as X	The bow lute, such as the Bambara ndang...

- Relations between specific entities
  - located-in (ORGANIZATION, LOCATION)
  - founded (PERSON, ORGANIZATION)
  - cures (DRUG, DISEASE)



employs (ORGANIZATION, PERSON)
causes (DRUG, DISEASE)

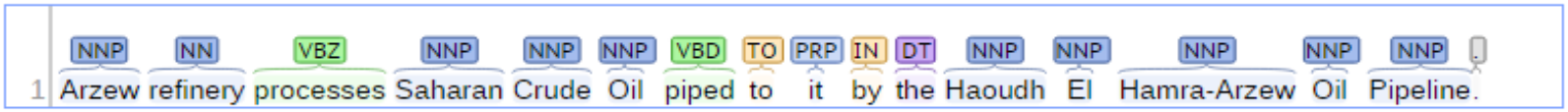
- Supply Chain :
  - direction matters!



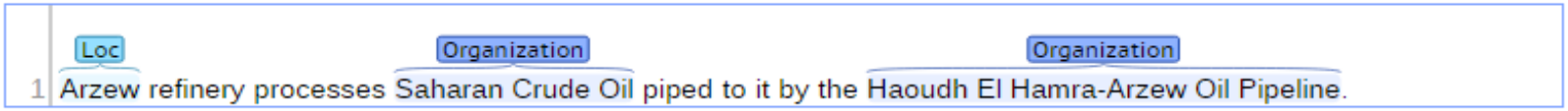
Stanford CoreNLP

Arzew refinery processes Saharan Crude Oil piped to it by the Haoudh El Hamra-Arzew Oil Pipeline.

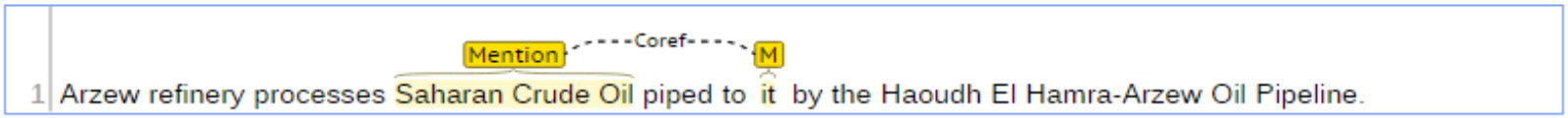
Part-of-Speech:



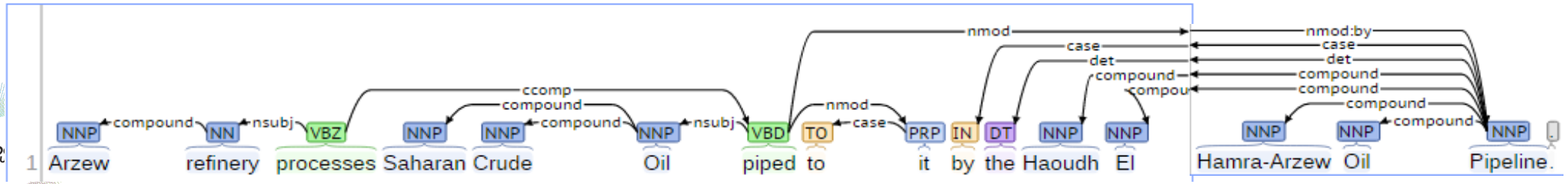
Named Entity Recognition:



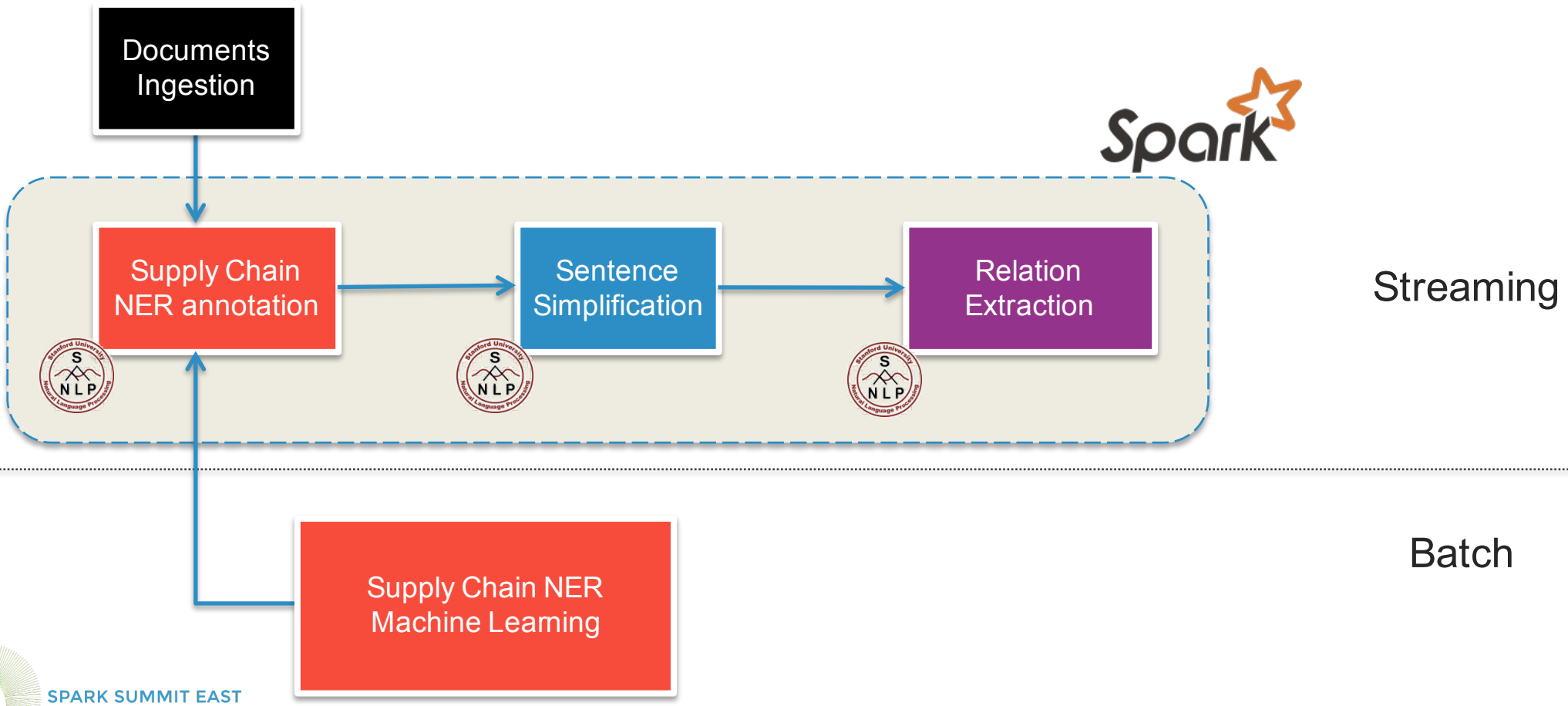
Coreference:



Basic Dependencies:



Supply Chain Relation Extraction Pipeline



Why Use Spark ?

1. Code reuse between batch layer & streaming processing layer
2. Easy to distribute Stanford NLP procesing
3. Spark brings the fault tolerance
4. Near Real Time is made easy for D.S and Developers compare to Apache Storm

Supply Chain Relation Extraction Pipeline

"Arzew refinery processes Saharan Crude Oil piped to it along the Haoudh El Hamra-Arzew Oil Pipeline."

Arzew refinery (REF)
Saharan Crude Oil (PROD)
Haoudh El Hamra-Arzew Oil Pipeline (PIPE)

```
val NERaccumulator = sc.accumulator(scala.collection.mutable.Map[String, String]())(new MapParam)

   textFile.mapPartitions { partitionOfRecords =>
       lazy val classifier      = NERPackage.NERUtils.setupNERecognizer()
       partitionOfRecords.map(x => NERPackage.NERUtils.getNER(x(1), classifier))
       .map( x                  => NERaccumulator.add(x))
   }
```

NER for Oil & Gas Supply Chain

NER
Machine
Learning

- Oil & Gas Sites & Products
  - PROD - product
  - TERM - terminal
  - OFI - Oil field
  - REF - refinery
  - PIPE - pipeline
  - O – other

- Feature Engineering :
  - useChunks = true
  - useNPGovernor = true
  - useLemmas
  - maxRight = 6
  - useClassFeature = true
  - useWordTag

Crude	PROD
oil	PROD
can	O
be	O
supplied	O
from	O
the	O
Mediterranean	LOC
Sea	LOC
by	O
the	PIPE
Janaf	PIPE
Crude	PIPE
Oil	PIPE
Pipeline	PIPE

PROD: Crude oil
LOC : Mediterenean Sea
PIPE : the Janaf Crude Oil Pipeline

NER for Oil & Gas Supply Chain

NER
Machine
Learning

- Linear chain Conditional Random Field (CRF) sequence models
  - Lafferty, McCallum, and Pereira (2001)
- CRF combines discriminative modeling and sequence modeling (robust to violation of iid assumption)
- A state in CRF can depend on observations from any (even future) state.
- Training process is lengthy

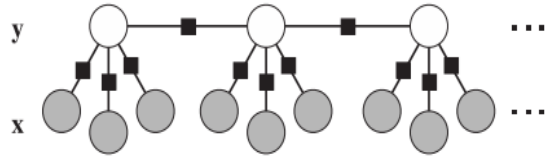


Figure 4.3 Graphical model of an HMM-like linear-chain CRF.

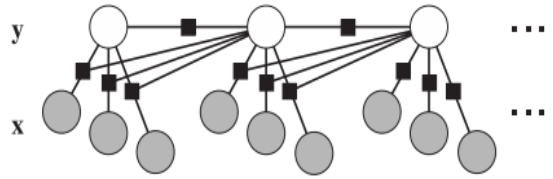


Figure 4.4 Graphical model of a linear-chain CRF in which the transition score depends on the current observation.

Image : Getoor L. and Taskar B, 2007 : Introduction to Statistical Relational Learning

Supply Chain Relation Extraction Pipeline

"Arzew refinery processes Saharan Crude Oil piped to it along the Haoudh El Hamra-Arzew Oil Pipeline."

1) *Arzew refinery processes Saharan Crude Oil.*

2) *Saharan Crude Oil is piped to it along the Haoudh El Hamra-Arzew Oil Pipeline.*

1) *supplier => none,*
receiver => Arzew refinery (REF)
theme =>Saharan Crude Oil (PROD)

2) *supplier => Haoudh El Hamra-Arzew Oil Pipeline (PIPE),*
receiver => arzew refinery (REF)
theme => Saharan Crude Oil(PROD)

```
import RelationExtractionPackage.{Relation, RelationUtils}
import SentenceSimplificationPackage.{SentenseSimplification, SentenceUtils}

val list_of_clauses = textFile.mapPartitions { partitionOfRecords =>

    lazy val relationExtractor: Relation              = RelationUtils.setupRelationExtraction()
    lazy val simplifier       : SentenseSimplification = SentenceUtils.setupSentenceSimplifier()

    partitionOfRecords.flatMap(SentenceUtils.simplify(_, simplifier))
    .map(clause => RelationUtils.getRelations(clause, relationExtractor, NERbroadcast.value))
 }.saveAsTextFile("src/main/resources/perfresults1.txt")
```

Relation Extraction

Relation Extraction : active voice

- VerbNet : fulfilling-13.4.1: supply, provide, serve, resupply

Example	Frames	Our Approach
Refinery A sends oil to Terminal B	NP V NP {To} PP.recepient	Supplier (subj) V Theme (obj) {To} Recepient (nmod=« to »)
Refinery A presents Terminal B with oil	NP V NP {With} PP.theme	Supplier (subj) V Recepient {With} Theme (obj)
Refinery A delivers oil	NP V NP	Supplier (subj) V Theme (obj)

- VerbNet: obtain-13.5.2: receive, obtain, gain, retrieve, collect

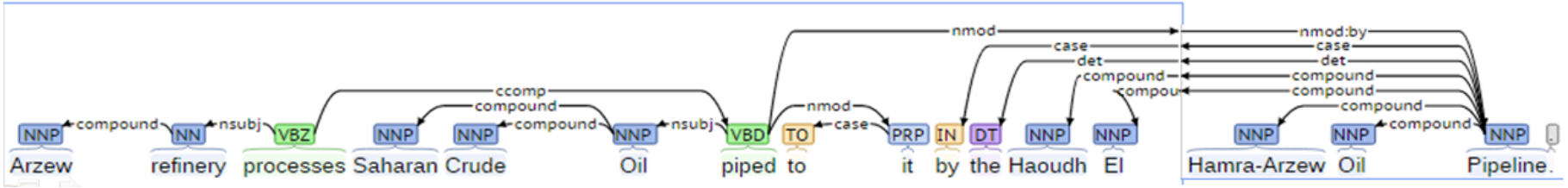
Example	Frames	Our Approach
Terminal B obtained oil	NP V NP	Recepient (subj) V Theme (obj)
Terminal B obtained oil from Refinery A	NP V NP {From} PP.source	Recepient (subj) V Theme (obj) {From} Supplier (nmod=« from »)

- Copula verbs : VerbNet seem-109-1-1: be, seem available

Example	Frames	Our Approach
Gas is available from Refinery A	NP V PP {From} NP	Theme (subj) V Attribute (available) {From} Supplier (nmod = « from »)

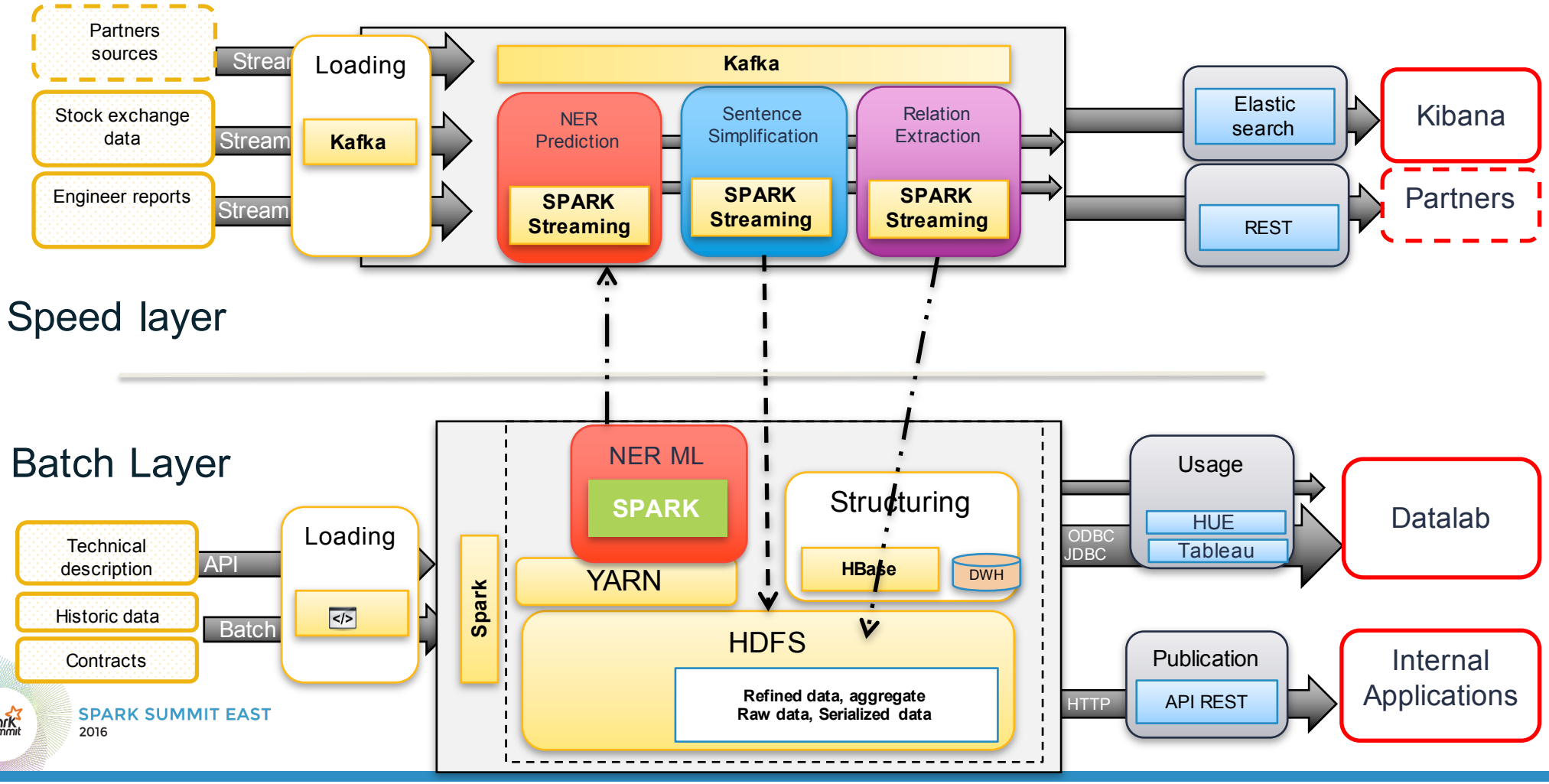
Sentence Simplification

Sentence Simplification



- Del Corro L.and Rainer Gemulla (WWW-2013) : *ClausIE: Clause-Based Open Information Extraction*
- Construct a clause for every subject dependency :
  - Clause: part of a sentence, that expresses some coherent piece of information
  - Consists of: Subject (S), Verb (V), Optionally: Indirect object (O), Direct object (O), Complement (C), one ore more adverbials (A)
- Replace relative pronoun (e.g., who or which) of a relative clause by its antecedent (relcl depend.).
- Generate clause from participial modifiers (ccomp, amod), which indicate reduced relative clauses.
- Result :
  - *"Arzew refinery processes Saharan Crude Oil."*
  - *"Saharan Crude Oil is piped to it along the Haoudh El Hamra-Arzew Oil Pipeline."*

Solution architecture



Streaming object reused

```scala
// messages is a dstream containing a flow of rdd
// each rdd has N+ (schema, sentence)
messages.foreachRDD { rdd =>
  rdd.mapPartitions{ partitionOfRecords =>
    lazy val relationExtractor : Relation = RelationUtils.setupRelationExtraction()
    lazy val simplifier : SentenseSimplification = SentenceUtils.setupSentenceSimplifier()
    // Parser Reuse between RDD
    partitionOfRecords.map{ x =>
      //sentence simplification
      val sentence = parseConnectMessage(x, 2)
      SentenceUtils.simplify(sentence, simplifier)
    }
      // Separating sentence simplification propositions
      .flatMap(_.split(","))
      .map(proposition => RelationUtils.getRelations(proposition, relationExtractor))
  }.saveToEs("RelationsExtraction") // end of mapPartition
} // end of foreachRDD

// Start the computation
ssc.start()
ssc.awaitTermination()
```

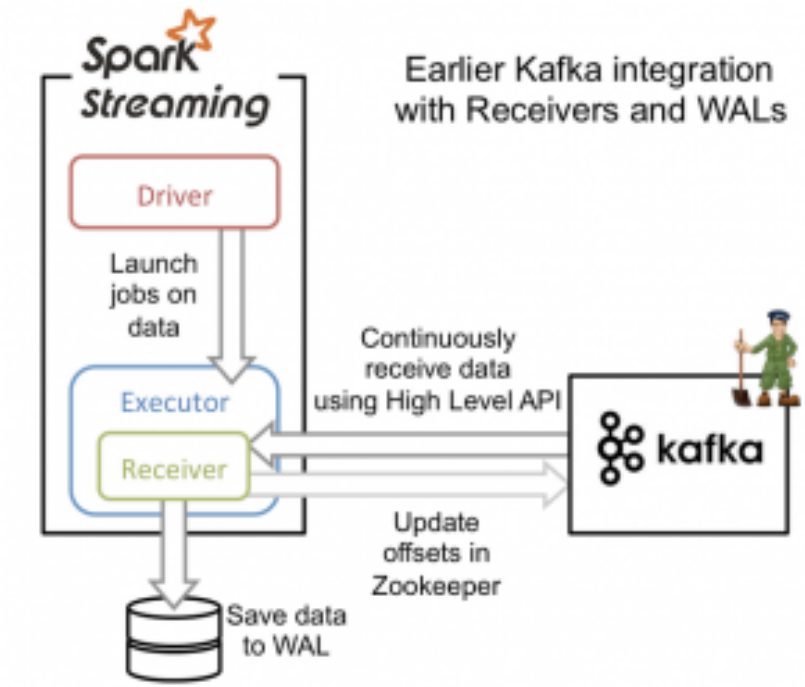
Reuse

```scala
object SentenceUtils {

  private [this] val simplifier = new SentenseSimplification()

  /**
    * Instantiate a Sentence Simplifier
    * @return
    */
  def setupSentenceSimplifier() : SentenseSimplification = {
    simplifier.initParser()
    simplifier
  }

  def simplify(sentence: String, simplifier : SentenseSimplification) : String = {
    simplifier.parse(sentence)
    simplifier.detectClauses()
    simplifier.generatePropositions()
    simplifier.getPropositions().asScala.mkString(", ")
  }
}
```

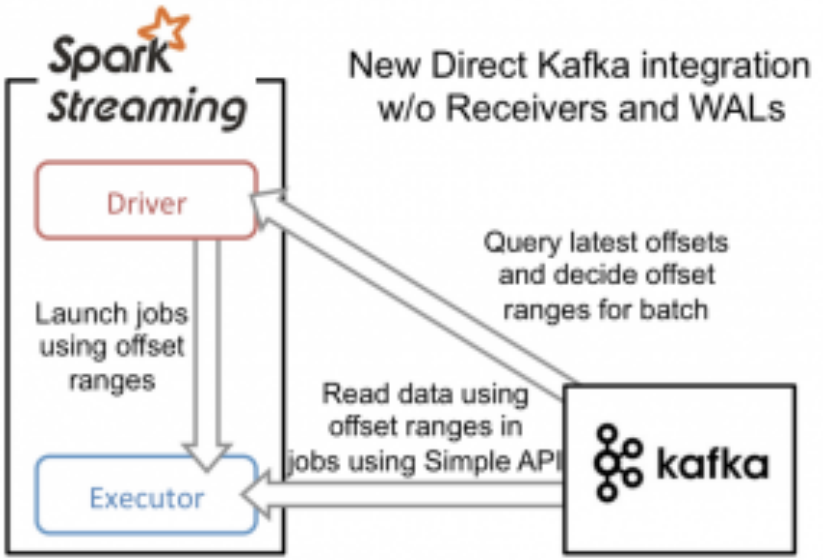
Annotators are time costly (3-4 seconds to initialize) so we initiate them only once per executor

Once processing & Fault tolerance

Receiver：At least once processing



Directstream : Exactly once processing



TAKE AWAY

Data Science

NER

- External libraries can be organically integrated in Spark (Spark Streaming) pipeline
- Distributed Spark CRF implementation would be very welcome

Relation Extraction

- Pattern-based implementation provides a good initial solution
- Next step : use it as training set for further bootstrap / ML

Sentence Simplification

- Very sensitive to changes in the Stanford Parser

ALL 3 blocks

- Share common Stanford NLP annotators. Dependency Parser is the most expensive.
- Nevertheless, implementation that reuses those in the tree blocks was found inefficient.
- Implementation with three consecutive maps in preferred.

Architecture & Tuning

Architecure

- Choose your kafka connector according to your existing monitoring tools
- Yarn scheduler resources calculator should take account of RAM & CPU (yarn.scheduler.capacity.resource-calculator)
- Do checkpoints

Memory

- Memory was a chokepoint
- Turn kryo serialization verify that you have registerd all your class spark.kryo.registrationRequired
- Use fastutils
- Filter as much as possible

Tuning

- Tune your partitions according to data volume
- Use coalesce instead of repartion if you are downgrading your partitions number
- Prefer Dataframe to RDD for Batch processing

And, of course, Capgemini is hiring

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

THANK YOU.

Yana Ponomarova : yana.ponomarova@capgemini.com
@yponomarova
Nicolas Claudon : nicolas.claudon@capgemini.com
@nicolasclaudon

SPARK SUMMIT EAST
DATA SCIENCE AND ENGINEERING AT SCALE
FEBRUARY 16-18, 2016 NEW YORK CITY