BRIDGE CONSULTING
INFORMATION TECHNOLOGY

Text Processing with Hadoop and Mahout
Key Concepts for Distributed NLP

- Based in Florence, Italy
- Founded in 1998
- 98 employees
- Business Areas
  - Retail, Manufacturing and Fashion
  - Knowledge Management
  - SOA Integration projects
  - BI and Analytics solutions
  - DBA services and support

ORACLE Gold Partner

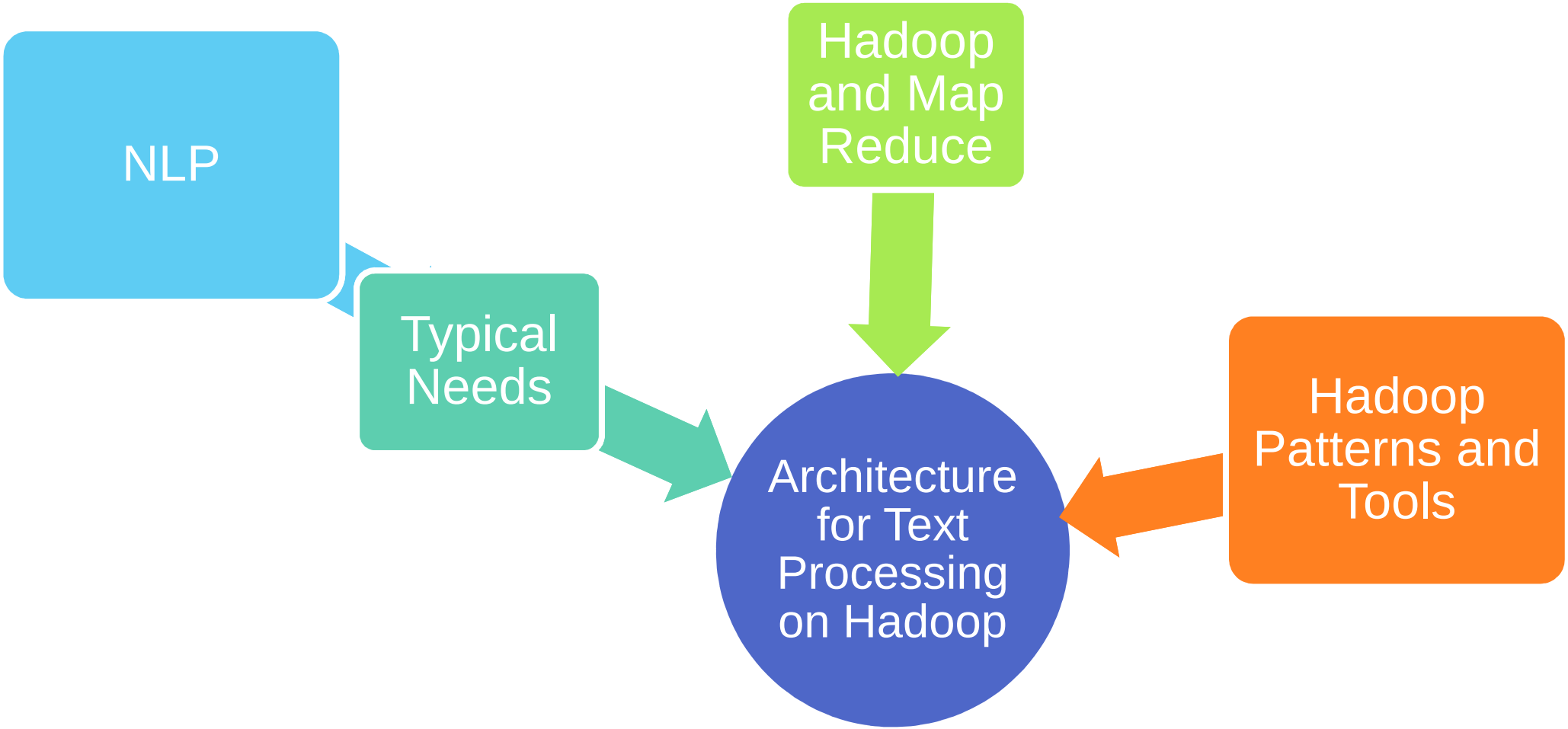
Manuel Zini

Distributed Systems and Emerging Technologies Team Leader@BridgeConsulting

Managing Hadoop Projects to build corporate knowledge management and search systems

Gained a PhD analyzing patent text into invention graphs (automatically ☺)

Agenda


NLP
Typical Needs
Hadoop and Map Reduce
Architecture for Text Processing on Hadoop
Hadoop Patterns and Tools

- Written natural language text is an enormous potential resource of information.
- Information lies in **e-mails, company documents, web pages, discussion forums and blogs, customer feedbacks, clinical records, patents, resumes, facebook pages, tweets** and new sources add everyday.
- The task of **managing**, **finding** and **extracting** information from text as already become unfeasible to be done manually or with traditional technologies.
- The ability to extract information from text is demanded from the market to increase **effectiveness** and **competitivity.**

Why Hadoop

BRIDGE CONSULTING
INFORMATION TECHNOLOGY

Text Processing

- Processing large quantities of text requires a significant computing power.
- Think for instance to processing tweets for sentiment analysis.

Hadoop

- Map Reduce is a programming model for distributed computation.
- Apache™ Hadoop® is an execution framework for map reduce on clusters of commodity servers.

Processing text on Hadoop enables the analysis of large quantities of text in a feasible time frame

Hadoop and Map Reduce
A Quick Introduction

BRIDGE CONSULTING
INFORMATION TECHNOLOGY

Map Reduce

- MapReduce is a programming model, an abstraction for writing programs that run on clusters of machines.
- The execution framework takes care of distributing data and processing in the cluster, hiding these complexities to the programmer.
- It consists basically of two phases: the Map phase and the Reduce phase

The Map Phase

In the Map phase, some computation is applied over all input records in a dataset. Several map tasks run in parallel over splits of input data.

The Reduce Phase

The output is then grouped by key and fed to a second user-specified computation for aggregation.

Hadoop

- Hadoop[9] is an open-source implementation of an execution framework for map-reduce
- Fastly growing ecosystem of other frameworks and tools for distributed computing

Hadoop comprises

- A framework for writing and executing map reduce jobs
- HDFS: a distributed file system implementation that allows for distributed map reduce execution exploiting the data locality principle

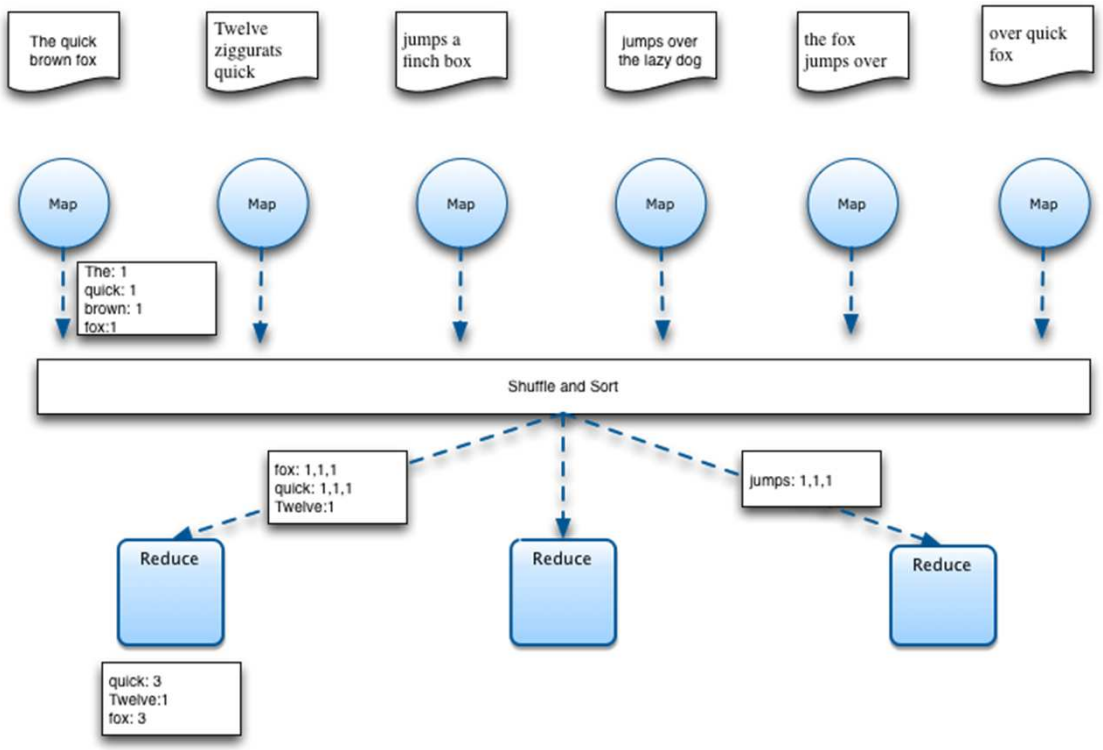
Mappers

- A MapReduce job is fed with data stored on the distributed file system and **splitted across nodes.**
- A map task is run for every split of the input file.
- The mapper is applied to every input key-value pair and outputs intermediate key-value pairs.

Reducers

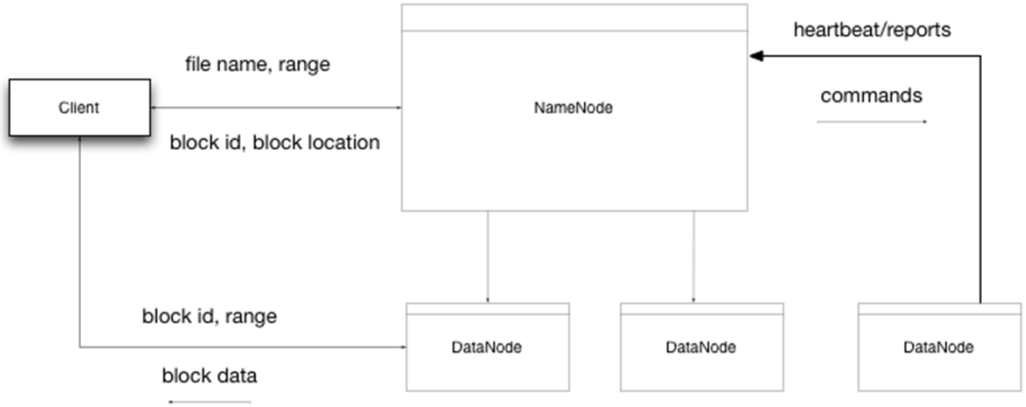
- A distributed ‘group by’ and sort operation is performed in the Shuffle & Sort phase. All key-value pairs for the same key are fed to the same reducer.[5]
- A reducer is applied to all values grouped for one key and in turn generates key-value pairs.
- Key-value pairs from each reducer are written on the distributed file system.

A complete job



HDFS – Hadoop Distributed File System

- File data is splitted into blocks.
- Blocks are stored and replicated in local disks of nodes in the cluster
- Master – Slave architecture



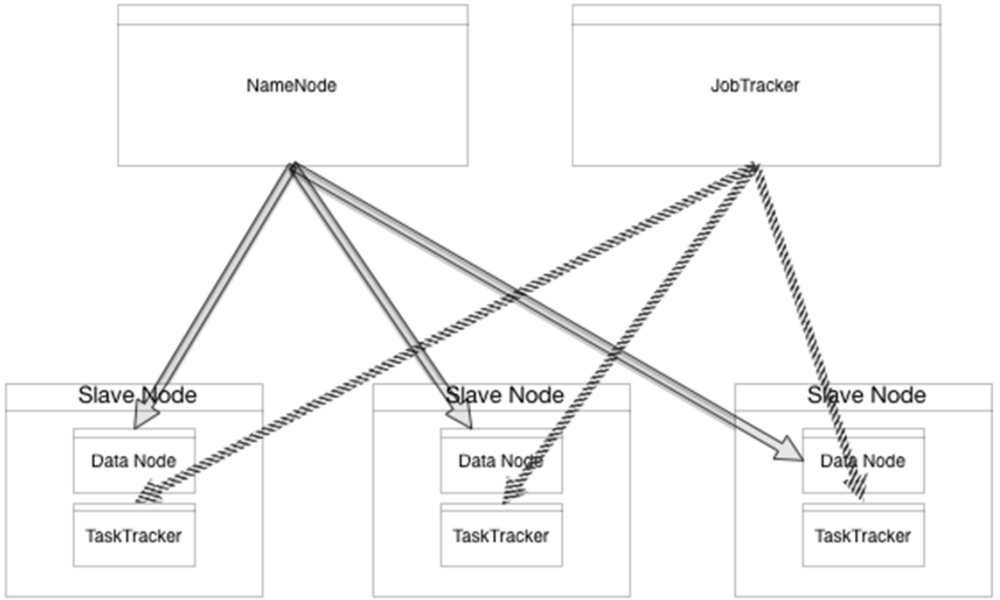
- Master keeps file system information (file to block mappings, block locations, metadata, directory structure, permissions)
- Slaves – host the replicated data blocks

BRIDGE CONSULTING
INFORMATION TECHNOLOGY

HDFS – Hadoop Distributed File System

- File splitting in blocks allows for distributed computation
- Each block can be processed by a different mapper
- The mapper is chosen to be co-located with data
- Principle of ‘data locality’: computation is moved close to data

Overall architecture of a cluster

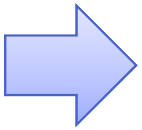


Natural Language Processing
a bird's eye view

NLP is the field of AI that aims at extracting information and, ultimately, knowledge from natural language.

Applications of natural language text processing

- Text mining
  - –Document clustering, statistical analysis of text…
- Information and Knowledge extraction
  - –Structured information
  - –Semantic networks
- Information Retrieval and Classification
  - –Findability of relevant information
- Sentiment analysis
  - –Analyze sentiment of text for customer relationship management
- Automatic translation
- Automatic summarization
- …



While different in purpose and techniques all these applications benefit from the tools developed by natural language processing

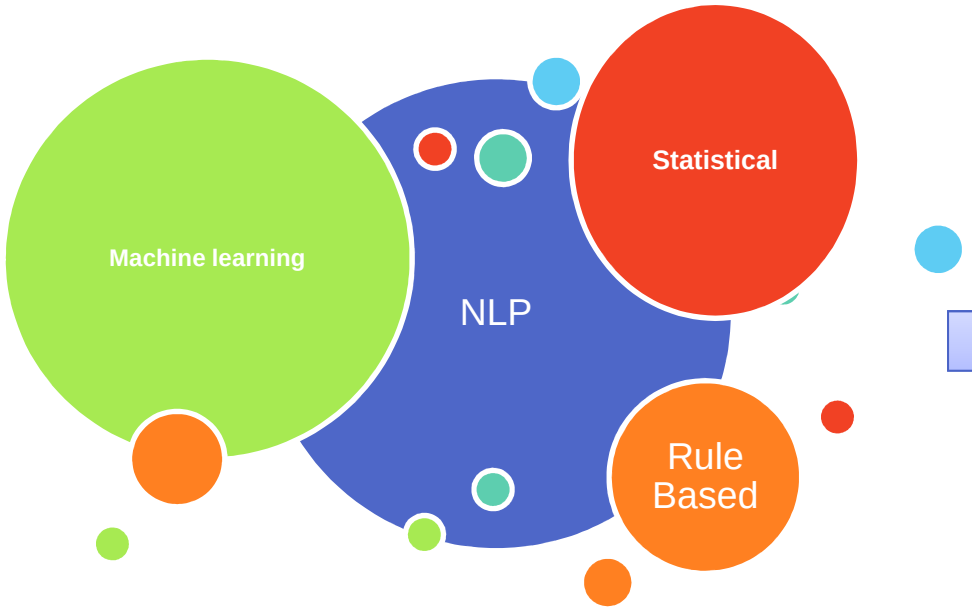
Text is unstructured

- False (or not completely true): text has some structure
- Language follows morphological, syntactical, and semantic rules.
- We have to identify available structure in text to extract information
- Information can then be structured the traditional way

A banana is yellow, weights one hundred grams and costs 20 cents.

item
type: banana
color: yellow
weight: 100 g
price: 20 cents

NLP Approaches

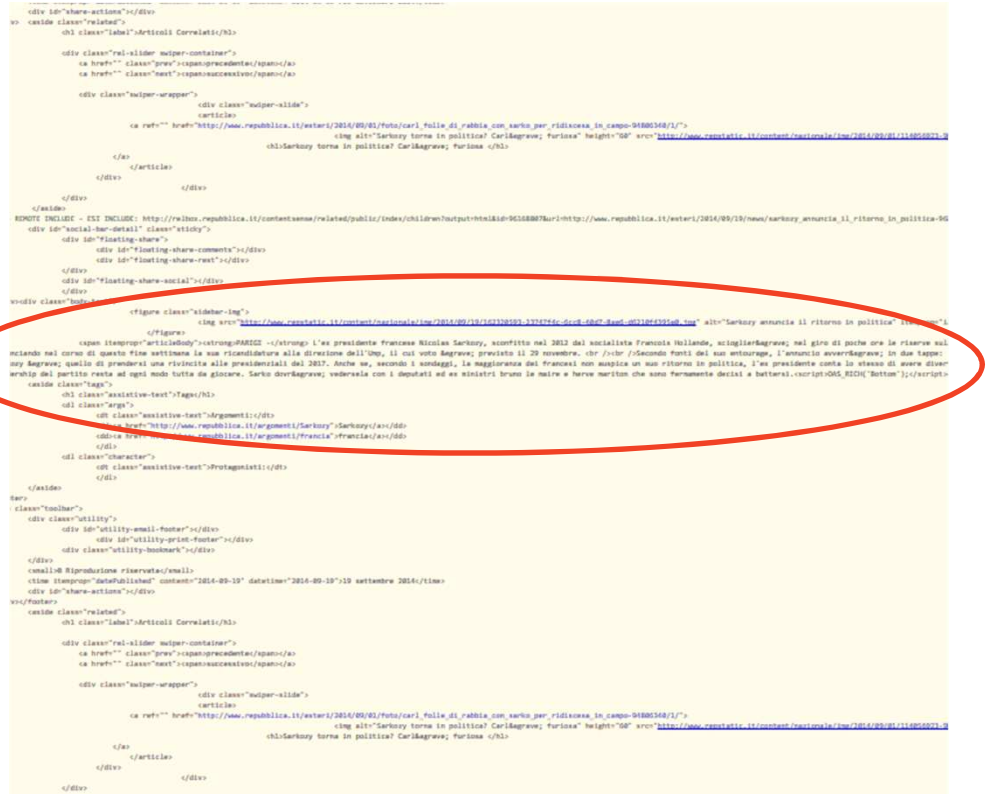


Different approaches to solve common NLP tasks:

- Text cleanup
- Language identification
- Tokenization
- Lemmatization/stemming
- Stop words filtering
- POS Tagging
- Keywords extraction/weighting
- Document classification

Text Conversion and Cleanup

- Text may come from HTML pages, PDFs, word documents, images and more
- It has to be converted to plain text
- Noise has to be removed (conversion errors, non printable or non alphabetic characters, spelling errors/abbreviations)
- In some case structure information may be retained (e.g. title/header information, bold or underlined text, metadata)



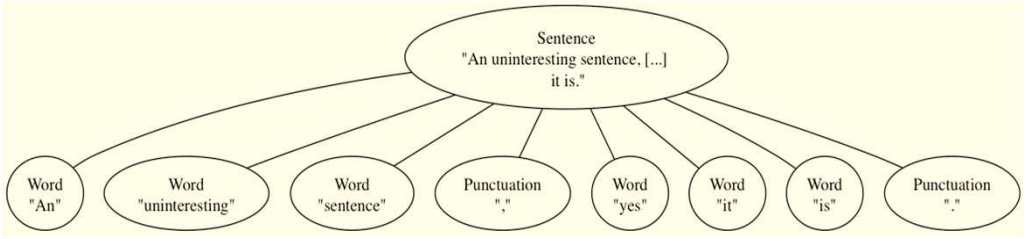
Language identification

- Text processing algorithms are language dependent: tokenization, stop words, lemmatization, POS tagging etc.
- Recognizing the language in which the text is written has to be performed early in the processing chain
- Language identification is the task of automatically identifying the language contained in a given document.

- Several methodologies: one should be chosen based on availability for the specific language, size of text, presence of foreign words.
- Statistical methods based on character strings n-grams (e.g. Naïve Bayes Classifier)
- Information theory methods (e.g. relative entropy in zip compression)
- Functional words frequencies (articles, pronouns, conjunction etc.)

Tokenization

- **Tokenization** is the process of splitting text into words, phrases and symbols.
- In the simplest case words boundaries can be recognized through whitespaces.
- This is not true for ‘San Francisco’.
- Some languages do not use spaces between words (e.g. Thai)… a different approach is required.
- German language allows composition: ‘Lebensversicherungsgesellschaftsangestellter’ means ‘employee of an insurance company’.



Transformation of inflected forms into a 'base' form.

- Stemming is a heuristic process, just removes affixes or postfixes following rules.
- Lemmatization tries to derive the 'base word' (lemma) using morphological rules, dictionaries and context.

Stemming:

biologically -> biologic
saw -> s

Lemmatization:

am, are, is, was -> be
have, has, had -> have
saw -> see or saw depending on context

Stop words filtering

- Stop words are words that will be removed before processing.
- Commonly performed to improve information retrieval or classification tasks (e.g. indexing and search, keyword extraction).
- Stop words are usually function words.
- Stop words may be application/task specific.

BRIDGE CONSULTING
INFORMATION TECHNOLOGY

Part Of Speech Tagging

- Tag each word with its part of speech (word classes like: adjective, noun, verb, preposition, conjunction, adverb etc.)
- Tag may include subcategories like: gender, number, tense etc.

```
<sentence id="0">
   <word wid="0" pos="DT">This</word>
   <word wid="1" pos="VBZ">is</word>
   <word wid="2" pos="DT">a</word>
   <word wid="3" pos="JJ">short</word>
   <word wid="4“pos="NN">sentence</word>
   <word wid="5" pos=".">.</word>
</sentence>
```

DT Determiner
VBZ Verb, 3rd person singular present
JJ Adjective
NN Noun, singular or mass

BRIDGE CONSULTING
INFORMATION TECHNOLOGY

Keyword extraction

- The task of extracting a meaningful subset of words that are able to describe the document's content.
- Usually extracted scoring terms or n-grams with tf-idf measure.

Term Frequency – Inverse Document Frequency

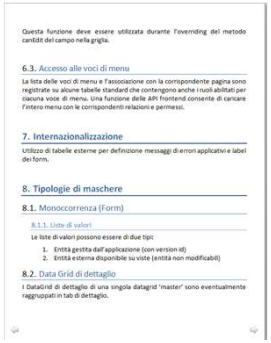
tf_k= term frequency of k in document

$$idf_k = \log\left(\frac{NDoc}{D_k}\right)$$

$$tf\text{-}idf = tf_k \cdot idf_k$$

Document classification

- The task of automatically assigning a document to a predefined category is called **supervised classification**
- Requires training of the classifier
- Examples:
  - Spam filtering is an example
  - Can be performed to enrich documents metadata with tags representing the category to improve findability


Questa funzione deve essere utilizzata durante l'overriding del metodo canEdit del campo nella griglia.

6.3. Accesso alle voci di menu

La lista delle voci di menu e l'associazione con la corrispondente pagina sono registrate su alcune tabelle standard che contengono anche i ruoli abilitati per ciascuna voce di menu. Una funzione delle API frontend consente di caricare l'intero menu con le corrispondenti relazioni e permessi.

7. Internazionalizzazione

Utilizzo di tabelle esterne per definizione messaggi di errori applicativi e label dei form.

8. Tipologie di maschere

8.1. Monoccorrenza (Form)

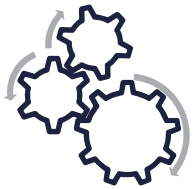
8.1.1. Liste di valori

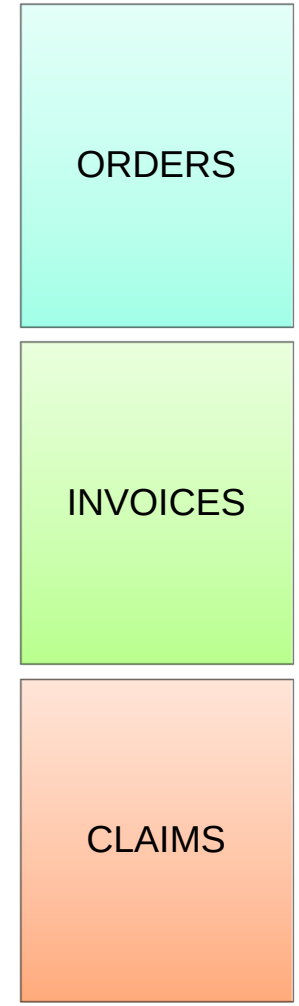
Le liste di valori possono essere di due tipi:

1. Entità gestita dall'applicazione (con version id)
2. Entità esterna disponibile su viste (entità non modificabili)

8.2. Data Grid di dettaglio

I DataGrid di dettaglio di una singola datagrid 'master' sono eventualmente raggruppati in tab di dettaglio.





Mahout

- Apache™ Mahout™ Project [8].
- A collection of machine learning libraries.
- By now supporting three major use cases:
  - Recommendations
  - Clustering
  - Document classification
- Most of the algorithms can be run on Hadoop Map Reduce.

Mahout algorithms

- Recommendations
  - User based recommenders, item-based recommenders
- Clustering
  - K-Means, Fuzzy K-Means, Spectral Clustering
- Document classification
  - Multinomial Naïve Bayes
  - Transformed Weight-Normalized Complement Naive Bayes implementation
  - Logistic Regression
  - HMM

Naive Bayes

- Transformed Weight-normalized Complement Naive Bayes
- Implemented as described in [2]
- Widely used and simple algorithm outperforming Naïve Bayes (MNB) on text documents
- Performance comparable to more complex algorithms (SVM)

Performances

	MNB	TWCNB	SVM
Industry Sector	0.582	0.923	0.934
20 Newsgroups	0.848	0.861	0.862
Reuters (micro)	0.739	0.844	0.887
Reuters (macro)	0.270	0.647	0.694

Improvements over Naïve Bayes [2]

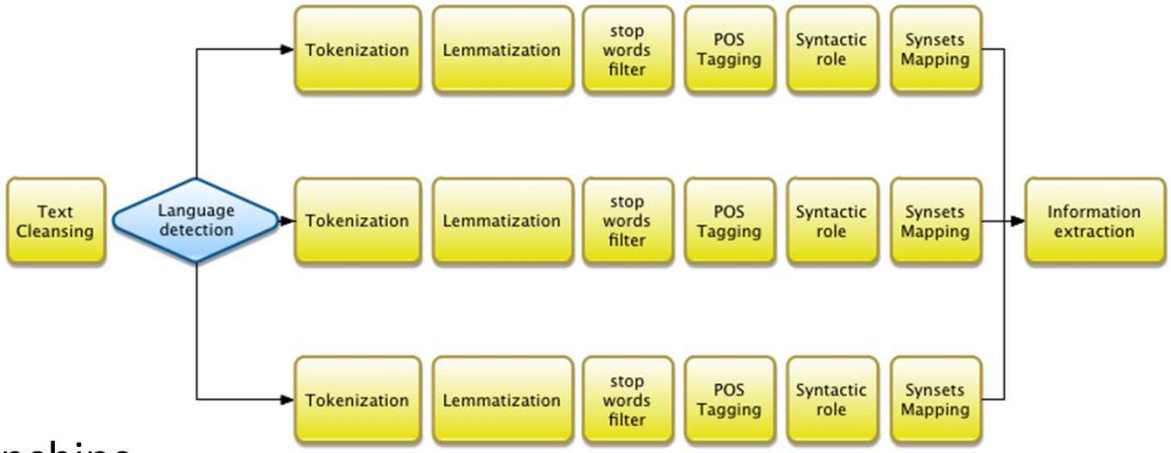
- TWCNB is more tolerant to training set data skew
- TWCNB normalizes for varying document length
- TWCNB makes internal use of TF-IDF weighting, lowering the importance in classification of common words
- It can be further improved by processing stages like stop words removal and stemming/lemmatization

- Four examples:

- Information extraction
- Keyword extraction
- Mahout TWCNB training
- Automated document classification

Information extraction example use case

- Task
  - –Extraction of a graph representing the description of a mechanical artifact

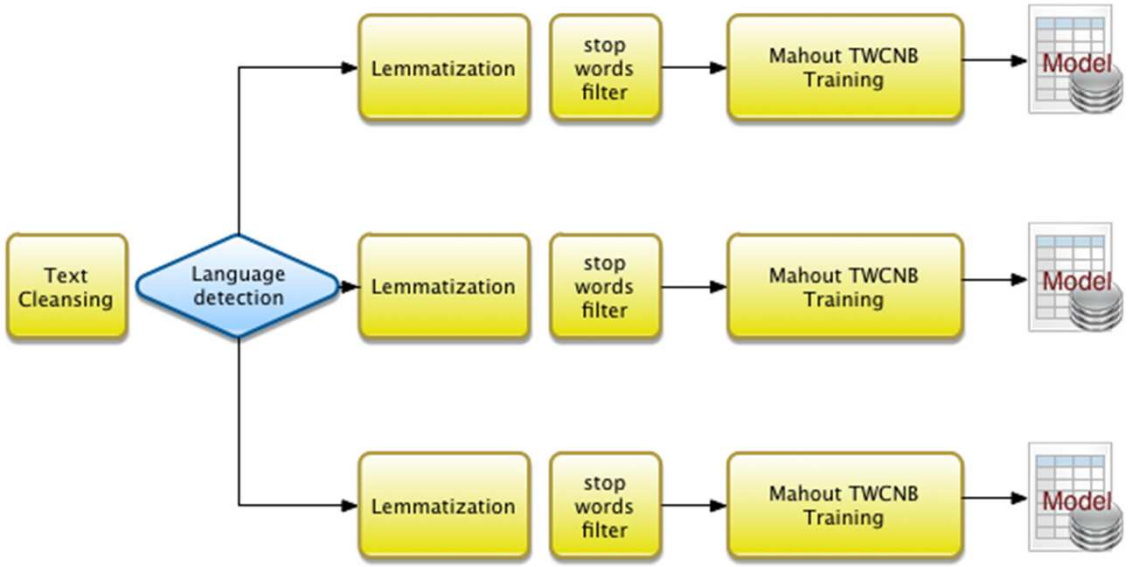


- E.g. an invention described in a patent
- The graph will describe compositional relationships and interactions between components

Document Classification: Training

Document classification training

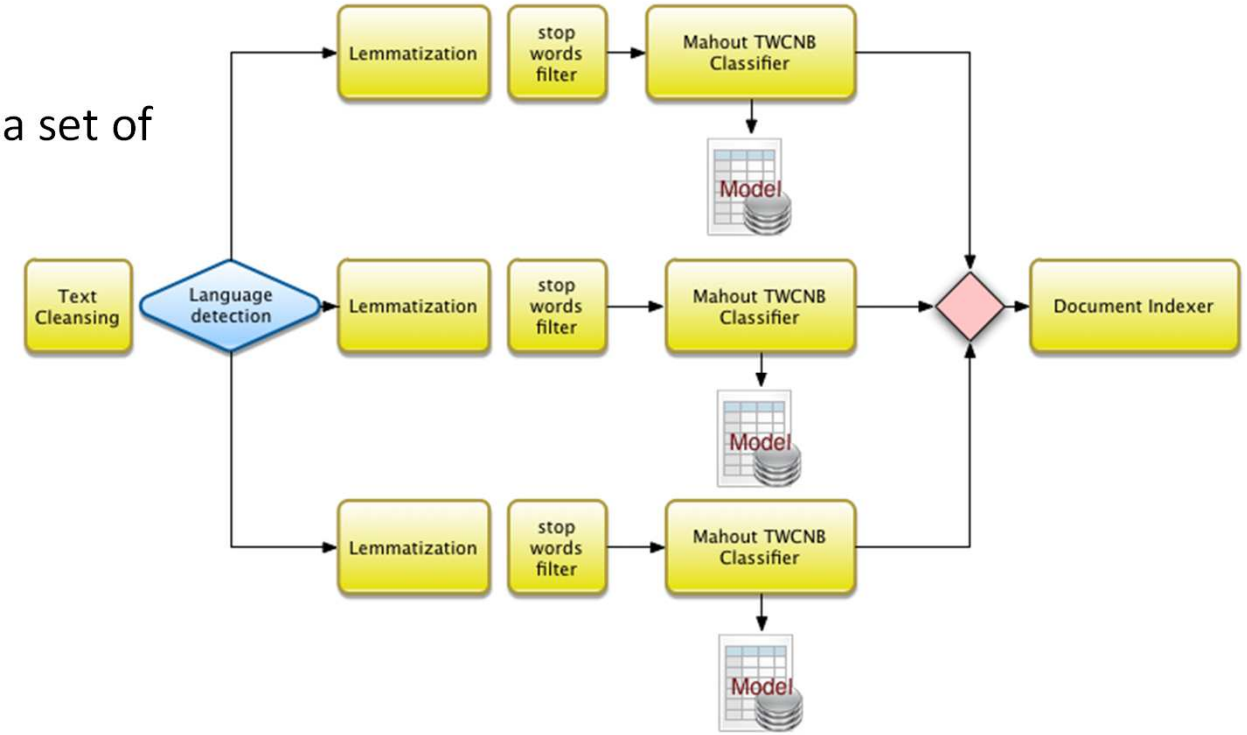
- Task
  - –Train the classifier
- Mahout TWCNB may be used
- One model will be generated for each language



Document Classification: Classification

Document classification example use case

- Task
  - –Classify each document in a class in a set of predetermined classes
- Mahout TWCNB may be used
- Classifier should be trained first

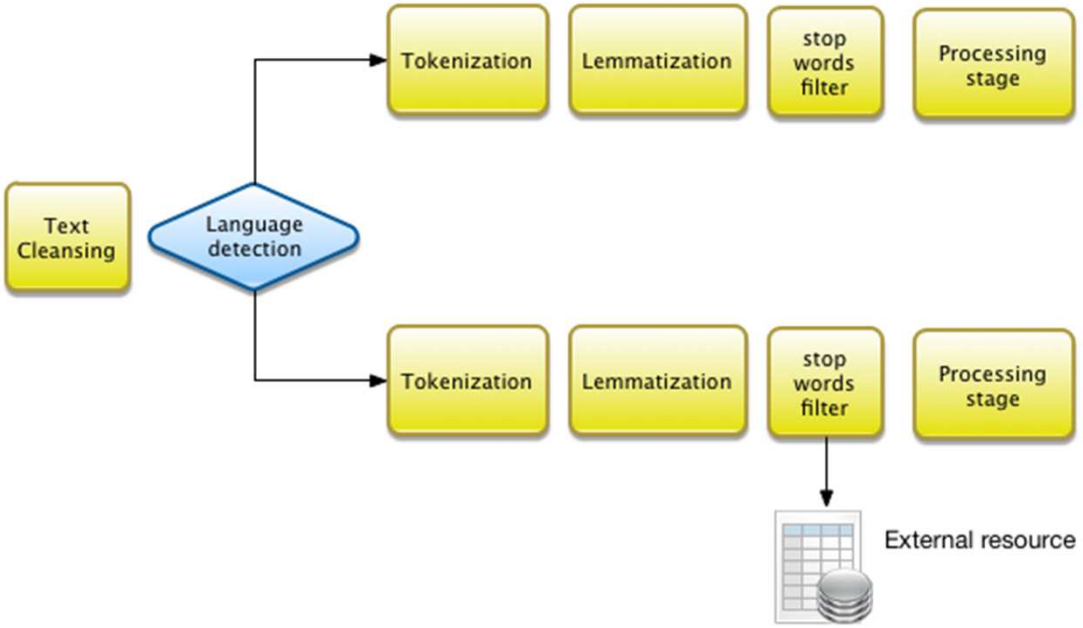


Keyword extraction example use case

- Task
  - –Extract relevant keywords from documents
- Keywords may be selected ranking words through TF/IDF measure

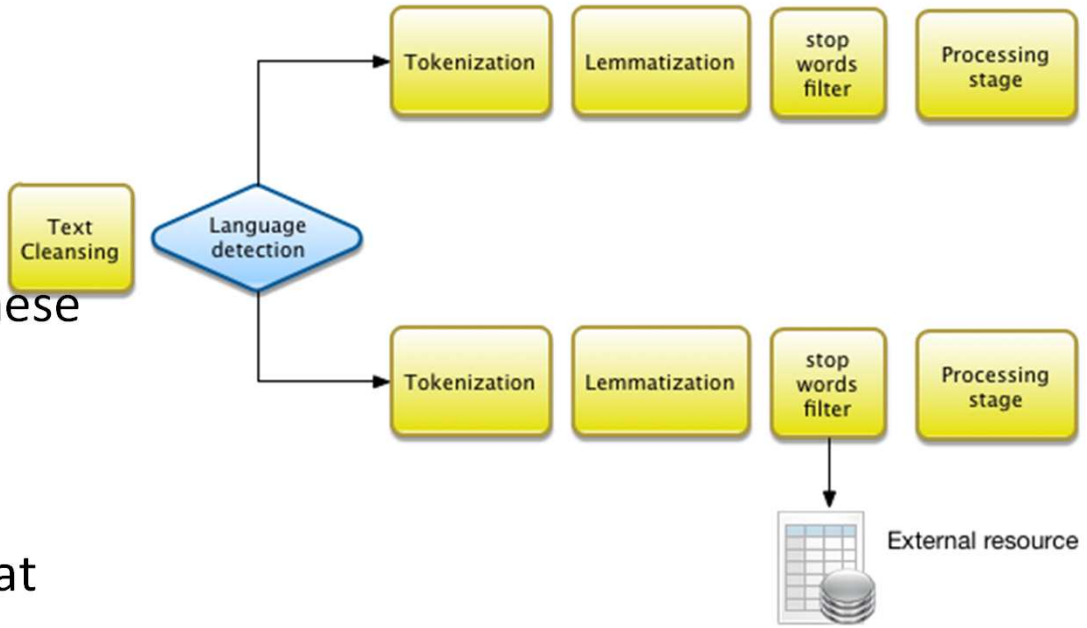
In a few text processing examples we've noticed some repeating patterns

- Processing happens in a pipeline of stages
  - The execution flow may change in response to some conditions
- Some of the processing stages are the same and could be reused
- Each use case has some specific processing stage
- Stages may use external resources



Processing pipeline

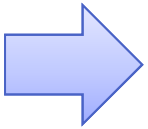
- The text processing pipeline is a frequent pattern in text processing
- Several text processing solutions organize these building blocks in a pipeline
- A key for processing blocks reuse is the definition of a common exchange data format



Executing NLP Pipelines on Hadoop

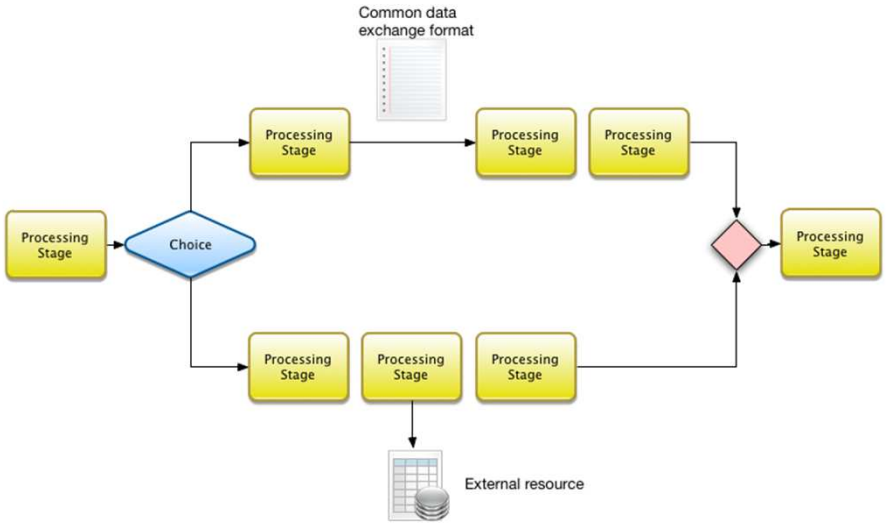
Common needs

Common needs emerged for processing text



- A processing pipeline solution with the ability to define conditional branches in the execution pipeline
- A mechanism for reuse of ‘processing stages’ and the composition of processing stages into pipelines.
- A data model and format that allows for interoperable data exchange between processing stages.
- The ability to make efficient use of external resources (dictionaries, graphs) or NLP libraries

Common needs

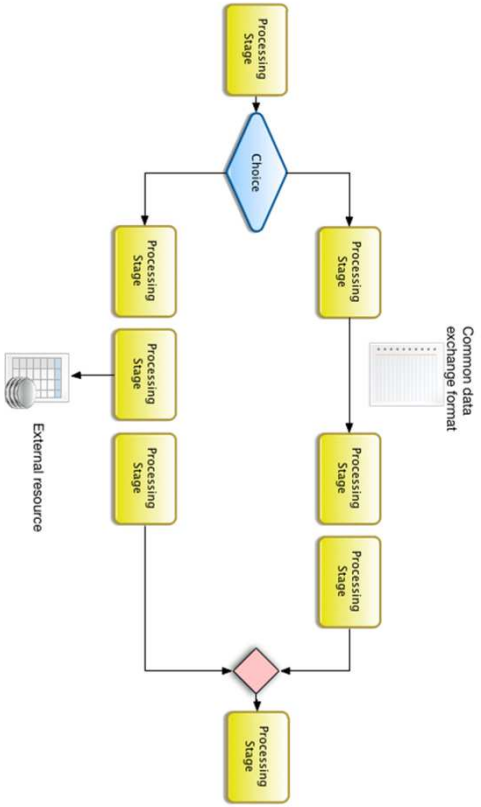


- A processing pipeline solution with the ability to define conditional branches in the execution pipeline
- A mechanism for reuse of ‘processing stages’ and the composition of processing stages into pipelines.
- A data model and format that allows for interoperable data exchange between processing stages.
- The ability to make efficient use of external resources (dictionaries, graphs) or NLP libraries

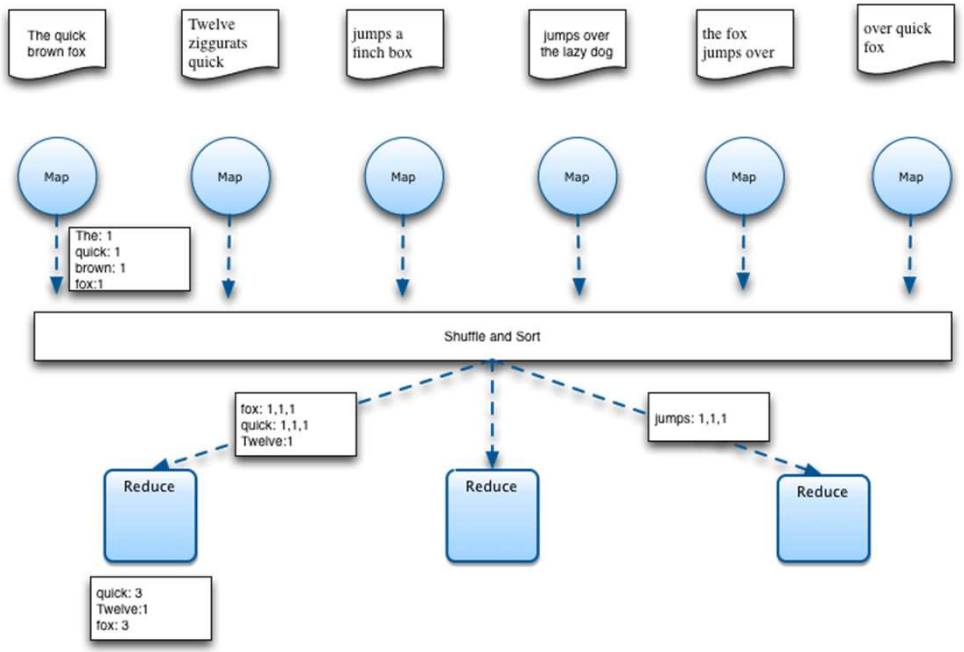
Mapping NLP needs on hadoop

BRIDGE CONSULTING
INFORMATION TECHNOLOGY

How can we map this…

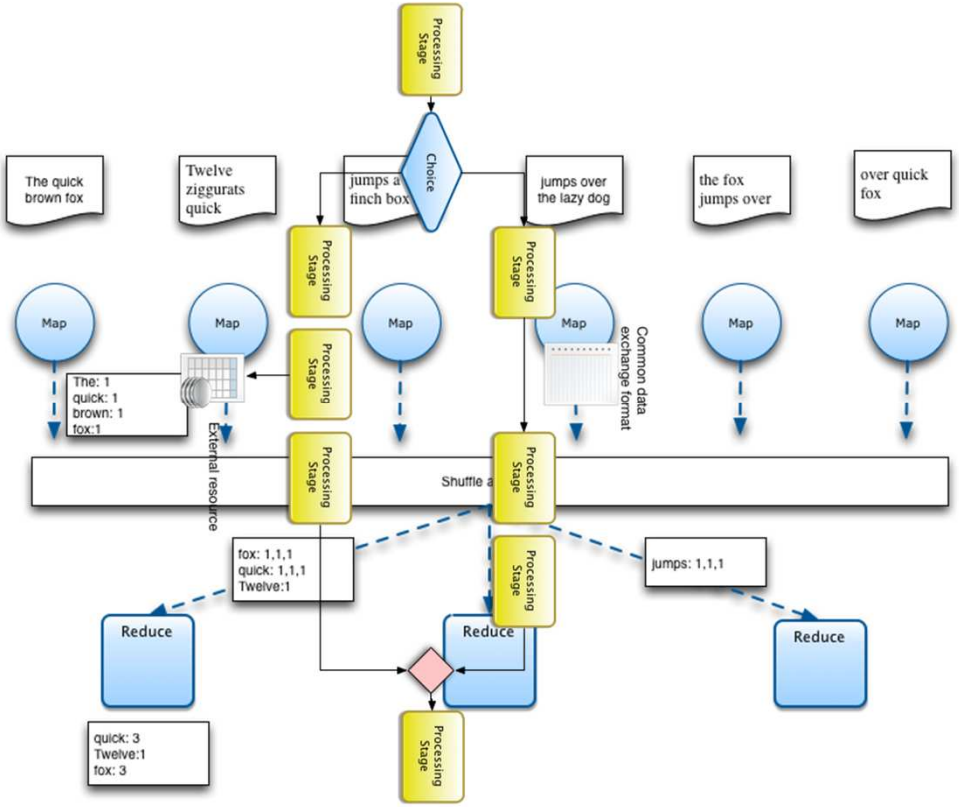


…onto this



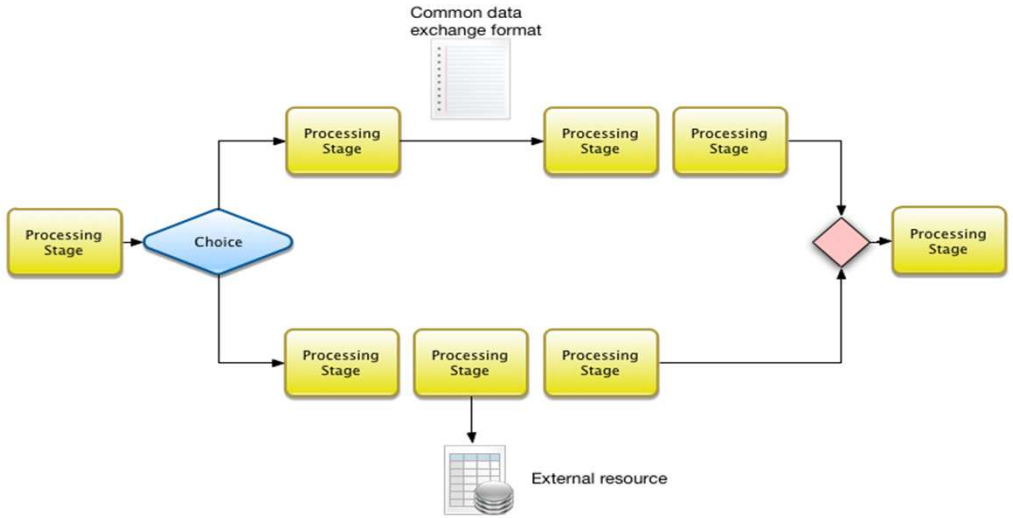
NLP and Hadoop

- Addressing the needs of text processing in a traditional non distributed programming environment is quite straightforward.
- **Addressing them on a distributed computing platform like Hadoop can be more challenging.**
- In order to successfully build NLP solutions on Hadoop a set of **viable solutions has to be identified.**



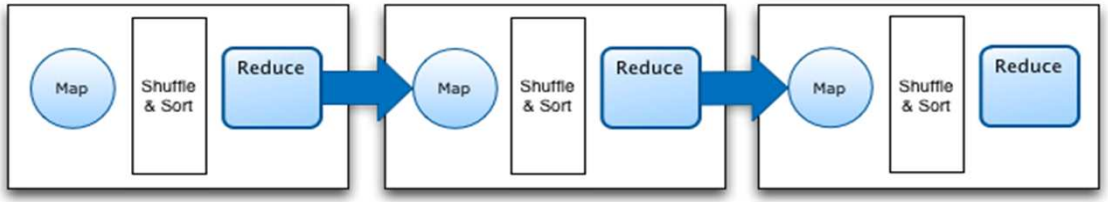
Problem

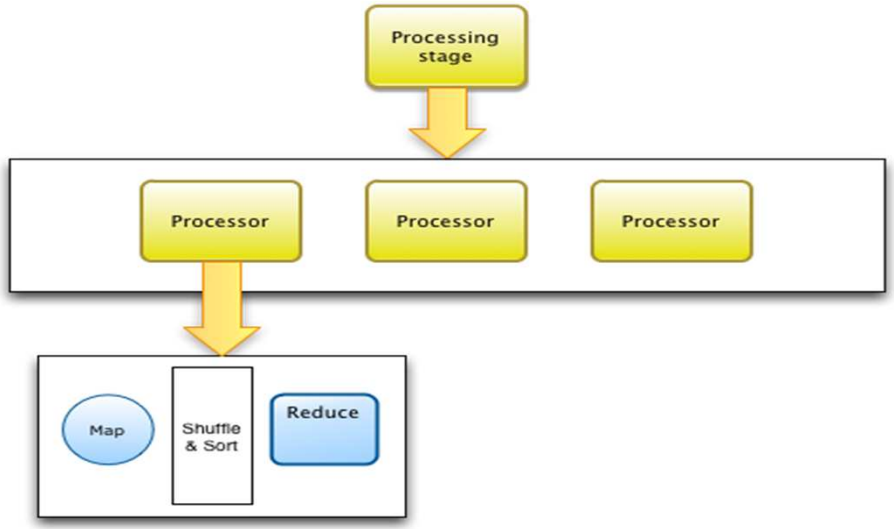
- A processing pipeline solution that allows for the distributed execution of subsequent processing stages.
- The pipeline should also support conditional branches in the execution flow.



Solution

- Map reduce Jobs may be executed in sequence.
- A processing stage may be modeled as a sequence of map-reduce jobs called processors.
- A workflow execution engine can be used to orchestrate the execution of processors and processing stages.
- There are several workflow execution engines available for Hadoop: Oozie, Azkaban, Luigi.





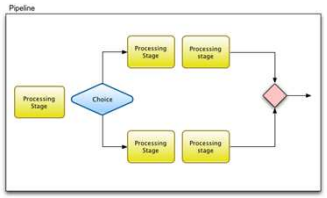
- Available or related patterns
  - This pattern has already been described in D.Miner *'MapReduce Design Patterns' [1]* as 'Job Chaining'
- Known implementations or related tools
  - The Oozie[10], Azkaban[12] and Luigi[13] frameworks address the problem of executing workflows of multiple jobs.

Oozie in short

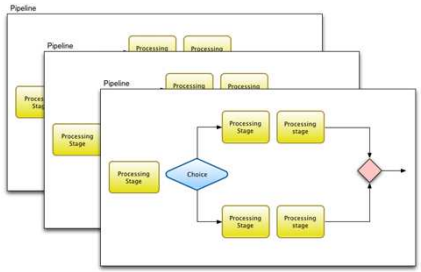
- Oozie [10] is a workflow scheduler system built to manage Apache Hadoop jobs.
- Workflows can be modeled through actions, decisions, forks and joins.
- Workflow actions are map-reduce jobs (or PIG, Hive, Shell, HDFS, Java etc.)
- Actions may also start child workflows
- Workflows are attached to coordinators that start the workflow when conditions are met
- Coordinators may be organized in bundles

Processor → Workflow action

Processing Stage → Sub Workflow



→ Workflow



→ Bundle

Pipeline Workflow example

```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="my-pipeline-wf">
<start to="subworkflow-stage-1"/>
<action name="stage-1-wf">
  <sub-workflow>
    <app-path>${nameNode}/${appDir}/wf-stage1.xml</app-path>
    <propagate-configuration/>
  </sub-workflow>
  <ok to="subworkflow-stage-2"/>
  <error to="fail"/>
</action>
<action name="stage-2-wf">
  <sub-workflow>
    <app-path>${nameNode}/${appDir}/wf-stage2.xml</app-path>
    <propagate-configuration/>
    <configuration>
      <property>
        <name>inputDir</name>
        <value>/${appDir}/data/output-1</value>
      </property>
      <property>
        <name>outputDir</name>
        <value>/${appDir}/data/output-2</value>
      </property>
    </configuration>
  </sub-workflow>
  <ok to="decision"/>
  <error to="fail"/>
</action>
<decision name="decision-node">
   <switch>
      <case to="stage-3-1">${fs:fileSize(outputFile) gt 10 * GB}</case>
        ...
      <case to="stage-3-2">${fs:fileSize(outputFile) lt 10 * GB}</case>
      <default to="end"/>
   </switch>
</decision>
…
<kill name="fail">
  <message>workflow failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
</kill>
<end name="end"/>
</workflow-app>
```

Processing Stage

```
<action name="stage-1-wf">
  <sub-workflow>
    <app-path>${nameNode}/${appDir}/wf-stage1.xml</app-path>
    <propagate-configuration/>
  </sub-workflow>
  <ok to="subworkflow-stage-2"/>
  <error to="fail"/>
</action>
```

Choice

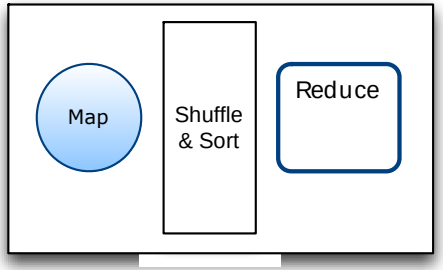
```
<decision name="decision-node">
   <switch>
      <case to="stage-3-1">${fs:fileSize(outputFile) gt 10 *
GB}</case>
        ...
      <case to="stage-3-2">${fs:fileSize(outputFile) lt 10 * GB}</case>
      <default to="end"/>
   </switch>
</decision>
```

Stage workflow example

Processing Stage

```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="stage-1-wf">
  <start to="processor-1"/>
  <action name="processor-1">
    <map-reduce>
      <prepare>
        <delete path="${nameNode}/${outputDir}"/>
      </prepare>
      <configuration>
        <property>
          <name>mapred.mapper.class</name>
          <value>Processor1Mapper</value>
          <name>mapred.reducer.class</name>
          <value>Processor1Reducer</value>
        </property>
      </configuration>
    </map-reduce>
    <ok to="processor-2"/>
    <error to="fail"/>
  </action>
  <action name="processor-2">
    <map-reduce>
      <prepare>
        <delete path="${nameNode}/${outputDir}"/>
      </prepare>
      <configuration>
        <property>
          <name>mapred.mapper.class</name>
          <value>Processor2Mapper</value>
          <name>mapred.reducer.class</name>
          <value>Processor2Reducer</value>
        </property>
      </configuration>
    </map-reduce>
    <ok to="end"/>
    <error to="fail"/>
  </action>
  <kill name="fail">
  <message>Map/Reduce failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
  </kill>
  <end name="end"/>
</workflow-app>
```

Processor




```
<action name="processor-1">
    <map-reduce>
      <prepare>
        <delete path="${nameNode}/${outputDir}"/>
      </prepare>
      <configuration>
        <property>
          <name>mapred.mapper.class</name>
          <value>Processor1Mapper</value>
          <name>mapred.reducer.class</name>
          <value>Processor1Reducer</value>
        </property>
      </configuration>
    </map-reduce>
    <ok to="processor-2"/>
    <error to="fail"/>
</action>
```

Processing stage reuse

Problem

- Processing stages should be designed and implemented as easily reusable and interoperable components.
- A stage should be be easily pluggable in different pipelines, this is not feasible without a common data structure and serialization format
- Serialized data marshalling/unmarshalling should be managed at framework level
- The file format should be easy to split for efficient distribution on the cluster

Processing stage reuse

Solution

- Processing stages should be packaged together to simplify reuse. Oozie workflow allows for the reuse of stages through the use of subworkflows.
- Processing stages should be pluggable in the pipeline where needed: a common exchange data model has to be defined that allows for the representation of text and text processing outcomes.
- This may range from a simple generic model to a more complex type system model.
- Use a compact serialization format, suitable for serializing the exchange data structure and possibly ‘split’ friendly: Avro may be an example

References

- Available or related patterns
  - A common model for analysis is already used for instance in UIMA™ (Unstructured Information Management Architecture) [6] and GATE (General Architecture for Text Engineering) [7]
- Known implementations or related tools
  - The UIMA CAS is an example of a common exchange model (Common Analysis System) for text processing
  - Apache™ Avro™ [11] is a serialization system for Map Reduce

UIMA in brief

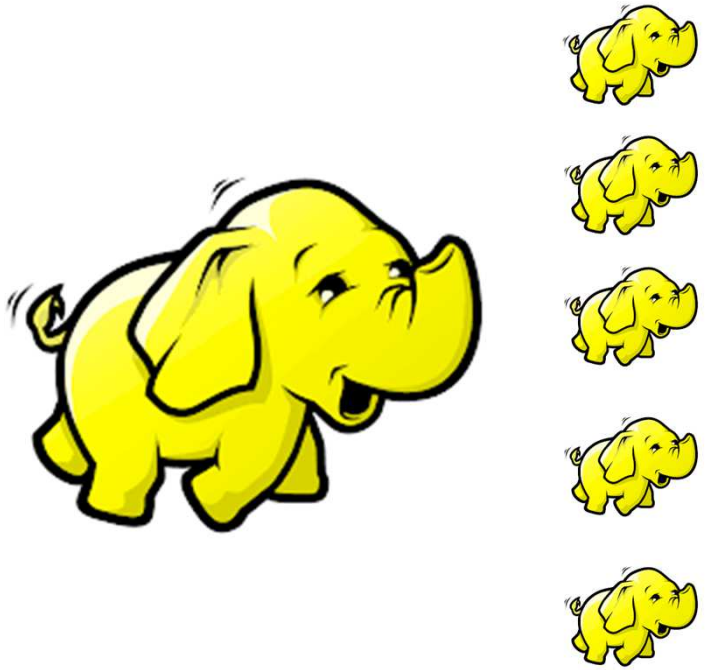
- Generalized framework for analyzing text (or other kinds of unstructured information)
- Component architecture aimed at maximizing component reusability and interoperability in the field of text processing
- All processing components in UIMA operate on a common data structure called CAS.
- A CAS document instance includes text (artifact), metadata and annotations generated by subsequent processing steps.
- Metadata generated by components may include a set of annotations that label regions of the document (or more generally of the ‘artifact’)
- Stand off model for annotations: annotations are separated from text.

AVRO in brief

- Apache™ Avro™ is a language-neutral data serialization system.
- Avro schemas are defined in JSON.
- Schema is stored with data in a serialized file.
- Compact format since no information needs to be stored with each field value.
- Object Container File Format: Compact file format, compressed in blocks.

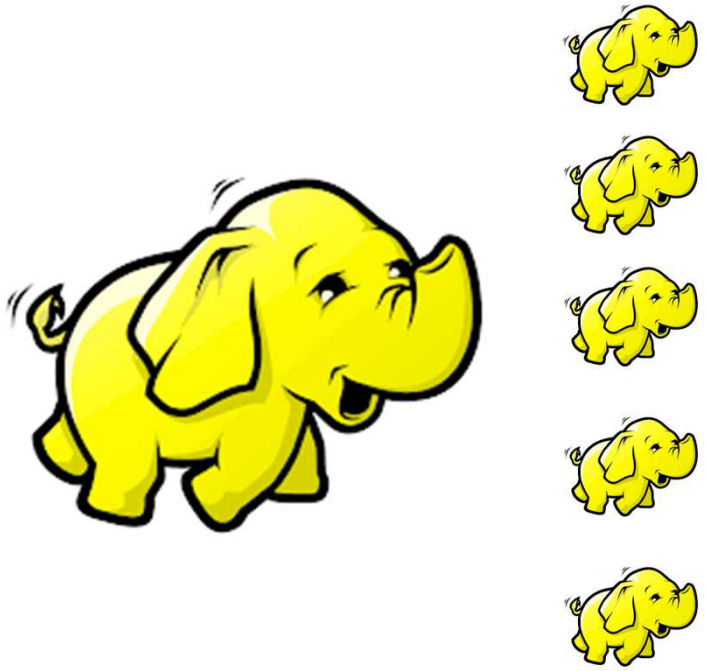
Problem

- Resources like dictionaries, frequency counts etc. are commonly used from processing stages.
- Libraries and tools are also needed in stages.
- Computation is distributed on potentially a great number of nodes.
- Resources and libraries should be available locally.



Solution

- Distributed Cache is a Hadoop facility for the automated and efficient distribution of read-only files.
- Compressed files and jars can be distributed on the cluster using the distributed cache.
- Jars can be added to the classpath of the task.



- Available or related patterns
  - Distributed cache is used for instance in D.Miner '*MapReduce Design Patterns' [1]* in the 'Replicated Join Pattern'.
- Known implementations or related tools
  - Distributed cache facility is already implemented in Hadoop

- Natural language processing is becoming both a need and an opportunity.
- Hadoop and Map Reduce are enablers for the mass processing of text.
- Some common needs have been identified for text processing.
- Solutions addressing those needs on Hadoop have been proposed, exploiting already existing patterns and tools to deploy NLP solutions on a distributed platform.
- Proposed solutions may be leveraged to design a generalized text processing architecture on Hadoop.

Bibliography

[1] Miner, Donald, and Adam Shook. *MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems*. " O'Reilly Media, Inc.", 2012.
[2] Rennie, Jason D., et al. "Tackling the poor assumptions of naive bayes text classifiers." *ICML*. Vol. 3. 2003.
[3] Belew, Richard K. "Finding out about." *A Cognitive Perspective on Search Engine Tech* (2000).
[4] Jurafsky, Daniel, and H. James. "Speech and language processing an introduction to natural language processing, computational linguistics, and speech." (2000).
[5]Lin, Jimmy, and Chris Dyer. "Data-intensive text processing with MapReduce." *Synthesis Lectures on Human Language Technologies* 3.1 (2010): 1-177.
[6] The Apache™ UIMA™ project. https://uima.apache.org/
[7] GATE – General Architecture For Text Engineering. https://gate.ac.uk/ . GATE is free software under the GNU licences and others.
[8] Apache™ Mahout™ - https://mahout.apache.org/
[9] Apache™ Hadoop® - http://hadoop.apache.org/
[10] Apache™ Oozie - http://oozie.apache.org/
[11] Apache™ Avro™ - http://avro.apache.org/
[12] Azkaban - http://azkaban.github.io/
[13] Luigi - https://github.com/spotify/luigi
[14] White, Tom. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

Q&A

Contact Info:

Manuel Zini
Team Leader
Distributed Systems and Emerging Technologies
mzini@bridgeconsulting.it