SCHEMA LEARNING IN APACHE SOLR

Abhishek Kumar Singh
Search Engineer, Unbxd

UNBXD

About Unbxd

E-Commerce Search, Discovery and Recommendation

Agenda

- Motivation
- Objectives for a new solution
- Schema Learning Framework
- Inference Modes
    - Primitives/Regex
    - Model
- Conclusion
- Roadmap

UNBXD

Motivation

We'll try to index the following 2 documents

	Title	Price	ProdId
Doc1	Deadpool 2	200	2017-03-01
Doc2	Fantastic Beasts	344.5	ssid-01-po

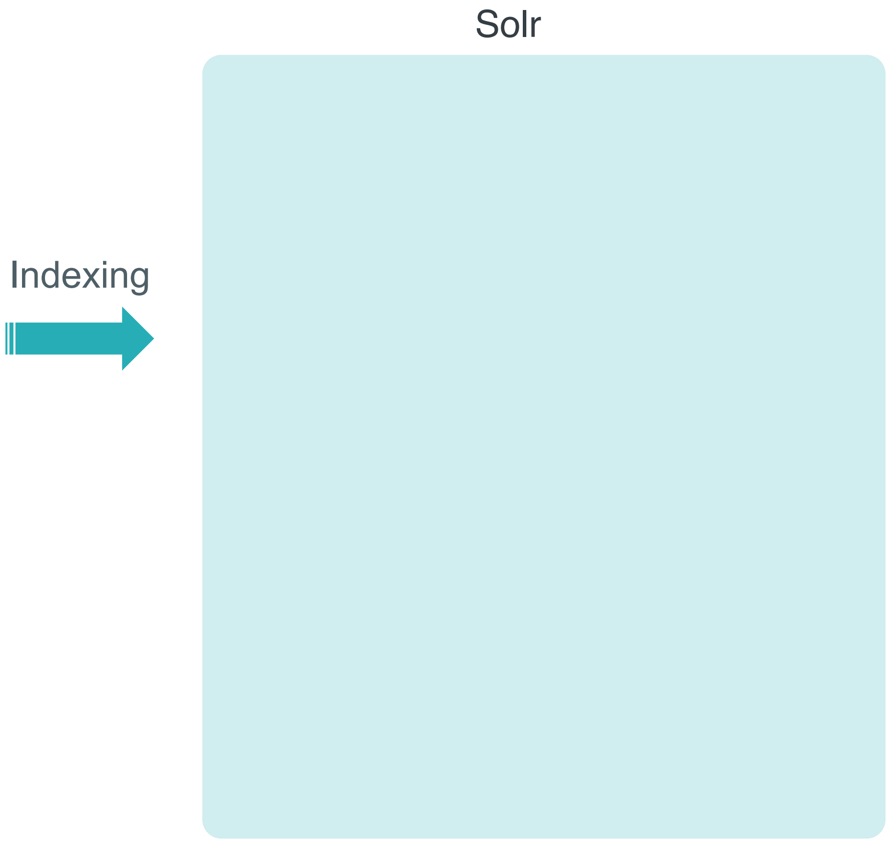
UNBXD

Motivation

Indexing 1st Doc:

	Title	Price	ProdId
Doc1	Deadpool 2	200	2017-03-01

	Title	Price	ProdId
Doc2	Beasts	178.77	ad23-33-a2



UNBXD

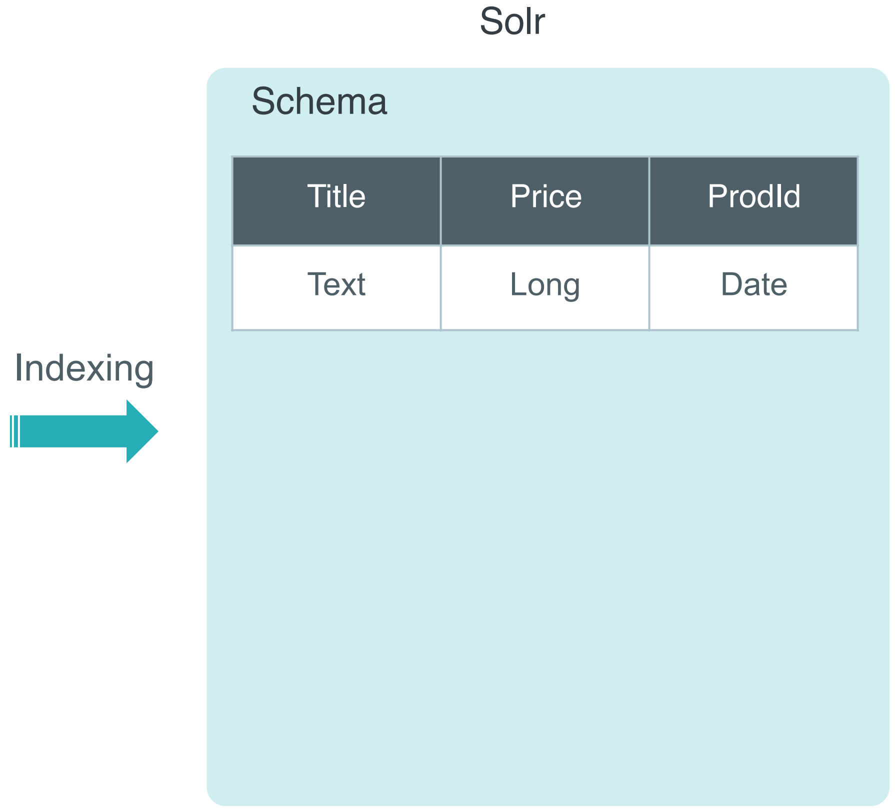
Motivation

Indexing 1st Doc:

Schema Inferred
Persisted to Managed Schema

	Title	Price	ProdId
Doc1	Deadpool 2	200	2017-03-01

	Title	Price	ProdId
Doc2	Beasts	178.77	ad23-33-a2



Title	Price	ProdId
Text	Long	Date

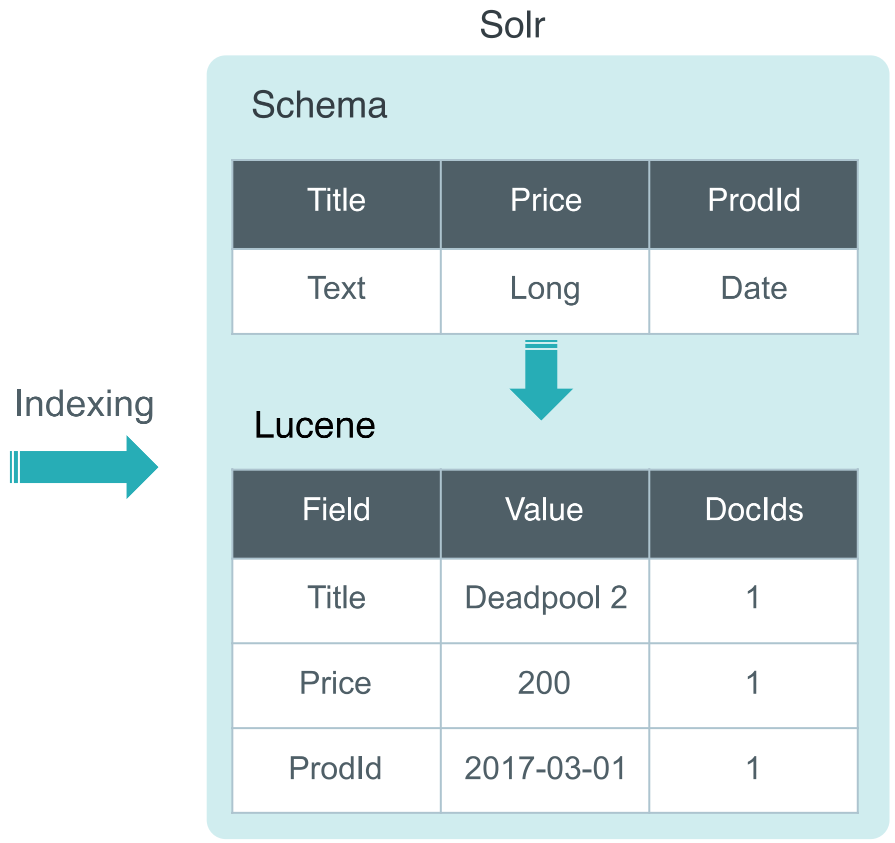
Motivation

Indexing 1st Doc:

Schema Inferred
Persisted to Managed Schema
Data Written to Lucene Index

	Title	Price	ProdId
Doc1	Deadpool 2	200	2017-03-01

	Title	Price	ProdId
Doc2	Beasts	178.77	ad23-33-a2



Title	Price	ProdId
Text	Long	Date

Field	Value	DocIds
Title	Deadpool 2	1
Price	200	1
ProdId	2017-03-01	1

UNBXD

Motivation

Indexing 2nd Doc:

Validation with previous Schema

	Title	Price	ProdId
Doc1	Deadpo	200	2017-03-01

	Title	Price	ProdId
Doc2	Beasts	178.77	ad23-33-a2

Indexing →

Solr

Schema

Title	Price	ProdId
Text	Long	Date

Lucene

Field	Value	DocIds
Title	Deadpool 2	1
Price	200	1
ProdId	2017-03-01	1

UNBXD

Motivation

Indexing 2nd Doc:

Validation with current Schema

-Validation Failed

	Title	Price	ProdId
Doc1	Deadpo	200	2017-03-01

	Title	Price	ProdId
Doc2	Beasts	178.77	ad23-33-a2

Indexing →

Solr

Schema

Title	Price	ProdId
Text	Long	Date

Lucene

Field	Value	DocIds
Title	Deadpool 2	1
Price	200	1
ProdId	2017-03-01	1

Error: **Schema Mismatch**

Expected Date Type in ProdId

UNBXD

Objectives

- Data driven schema generation
  - Schema is tightly coupled with data

- Infer optimal types
  - Compatible with all the docs

- Semantic type inference
  - **For Fashion E-Commerce:** Indentifying color, pattern and title types from catalogs.
  - **For AutoPart E-Com:** Indentifying year, make and model types from catalogs.

UNBXD

The Solution: Schema Learning Framework

- Schema generation without indexing

- Offline training
  - With raw documents
  - Support for Type Hierarchy configuration

- Generated Schema can be used for actual indexing

UNBXD

Schema Learning Framework

- Supports two types of inference :

❖ Primitives/Regex based inference

❖ ML Model based inference

UNBXD

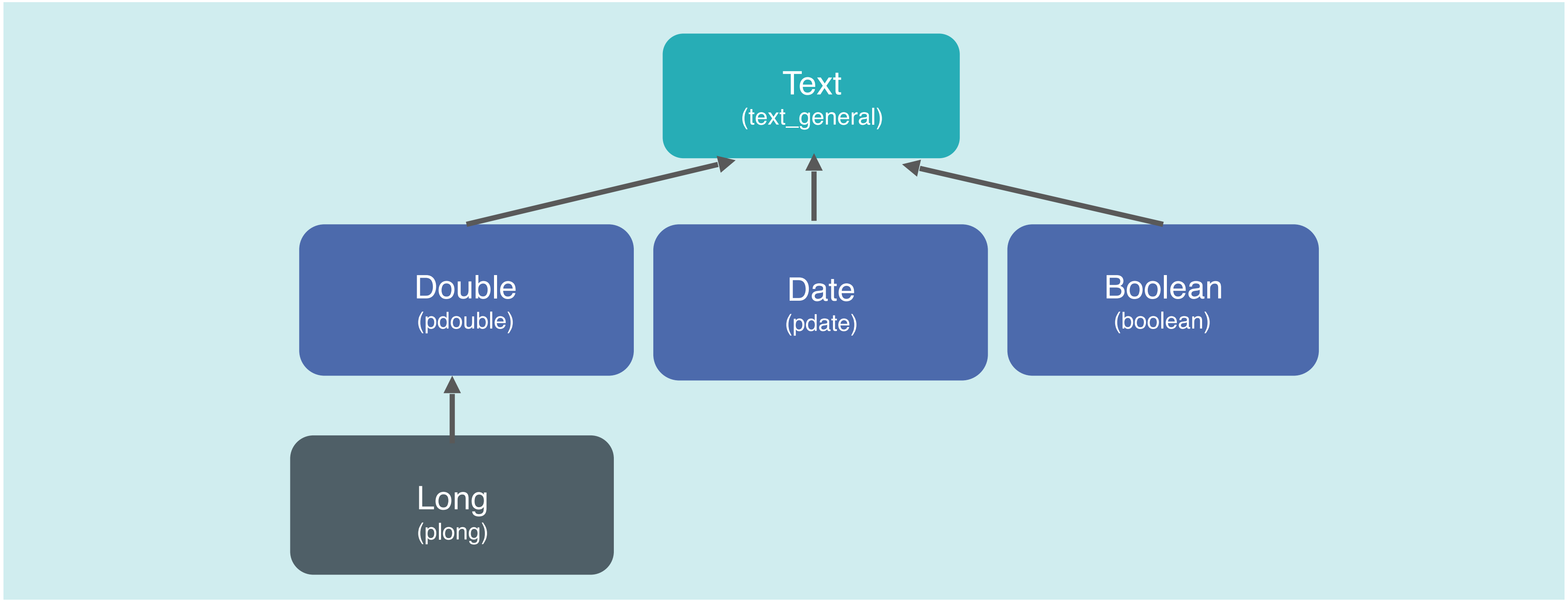
Schema Learning Framework:

Primitives/Regex based inference

- A Type Hierarchy needs to be defined for the Primitive/Regex Types
    - ❖ what other types a field type has to support
- This Type Hierarchy is used while Assigning optimal types
    - ❖ *Example:* A field having both *Long* and *Double* will be assigned *Double* Type
- Regex can be defined along side Primitive

Schema Learning Framework:

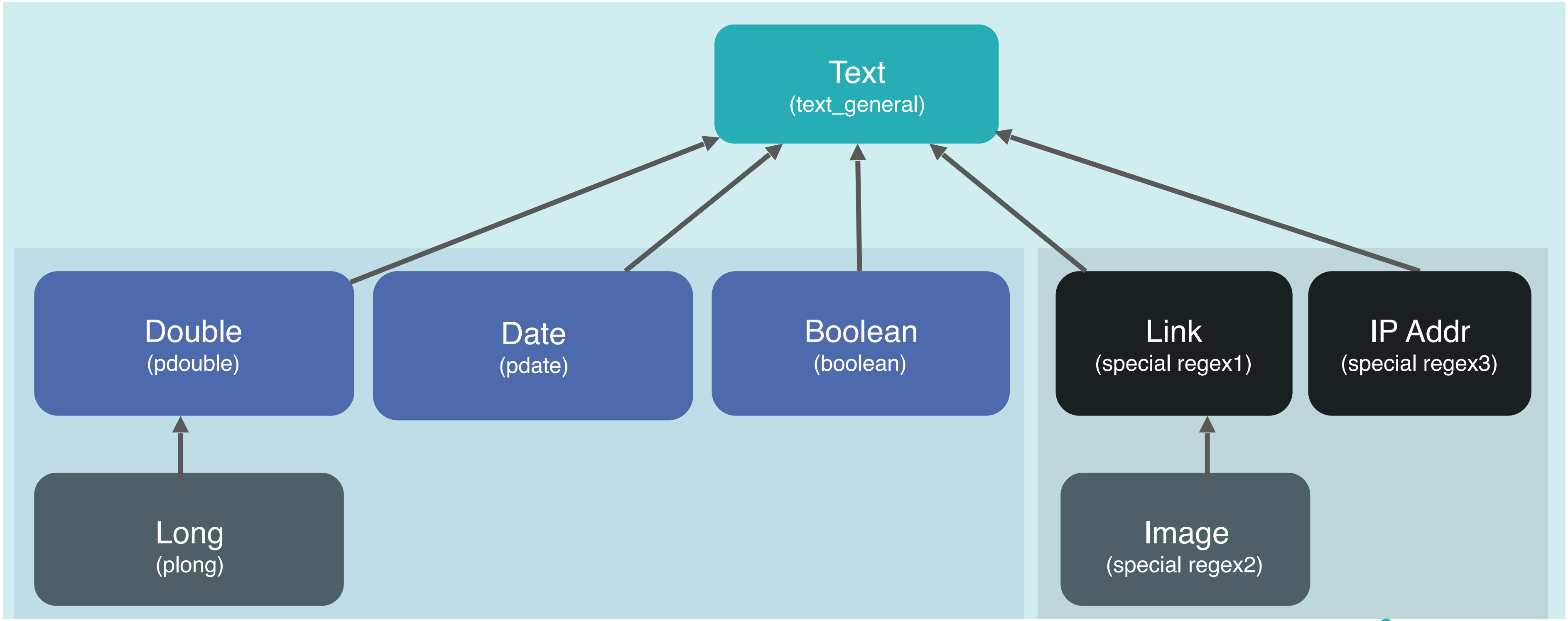
Type Hierarchy: Primitives based inference



UNBXD

Schema Learning Framework:

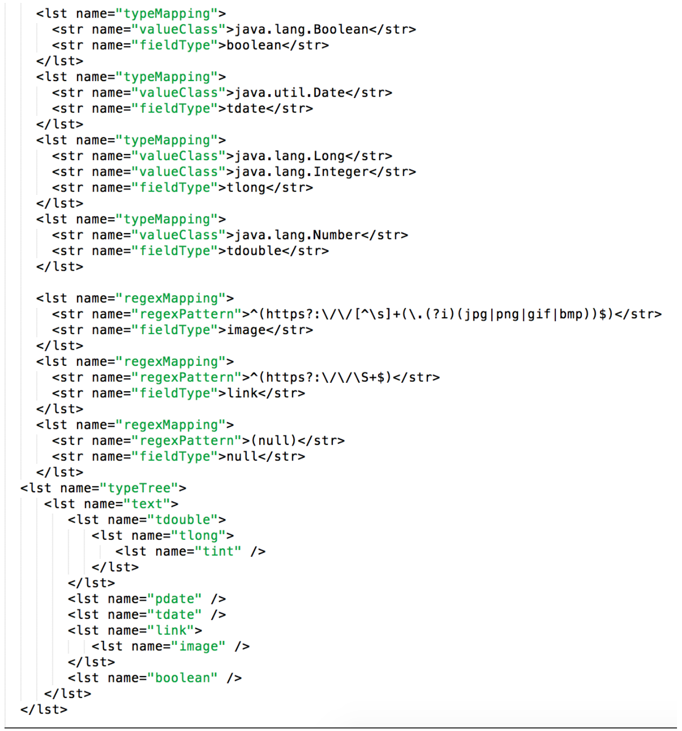
Type Hierarchy: Primitives+Regex based inference



UNBXD

Schema Learning Framework:

Primitives+Regex based inference : Configuration



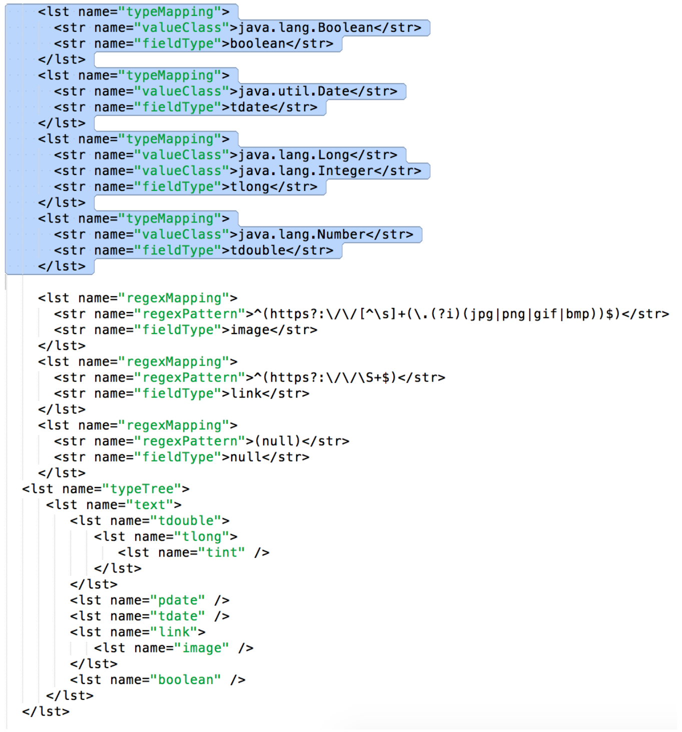
```
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Boolean</str>
    <str name="fieldType">boolean</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.util.Date</str>
    <str name="fieldType">tdate</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Long</str>
    <str name="valueClass">java.lang.Integer</str>
    <str name="fieldType">tlong</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Number</str>
    <str name="fieldType">tdouble</str>
  </lst>

  <lst name="regexMapping">
    <str name="regexPattern">^(https?:\/\/[^\s]+(\.(?i)(jpg|png|gif|bmp))$)</str>
    <str name="fieldType">image</str>
  </lst>
  <lst name="regexMapping">
    <str name="regexPattern">^(https?:\/\/\S+$)</str>
    <str name="fieldType">link</str>
  </lst>
  <lst name="regexMapping">
    <str name="regexPattern">(null)</str>
    <str name="fieldType">null</str>
  </lst>
<lst name="typeTree">
  <lst name="text">
    <lst name="tdouble">
      <lst name="tlong">
        <lst name="tint" />
      </lst>
    </lst>
    <lst name="pdate" />
    <lst name="tdate" />
    <lst name="link">
      <lst name="image" />
    </lst>
    <lst name="boolean" />
  </lst>
</lst>
```

UNBXD

Schema Learning Framework:

Primitives+Regex based inference : Configuring Primitive Types Mapping



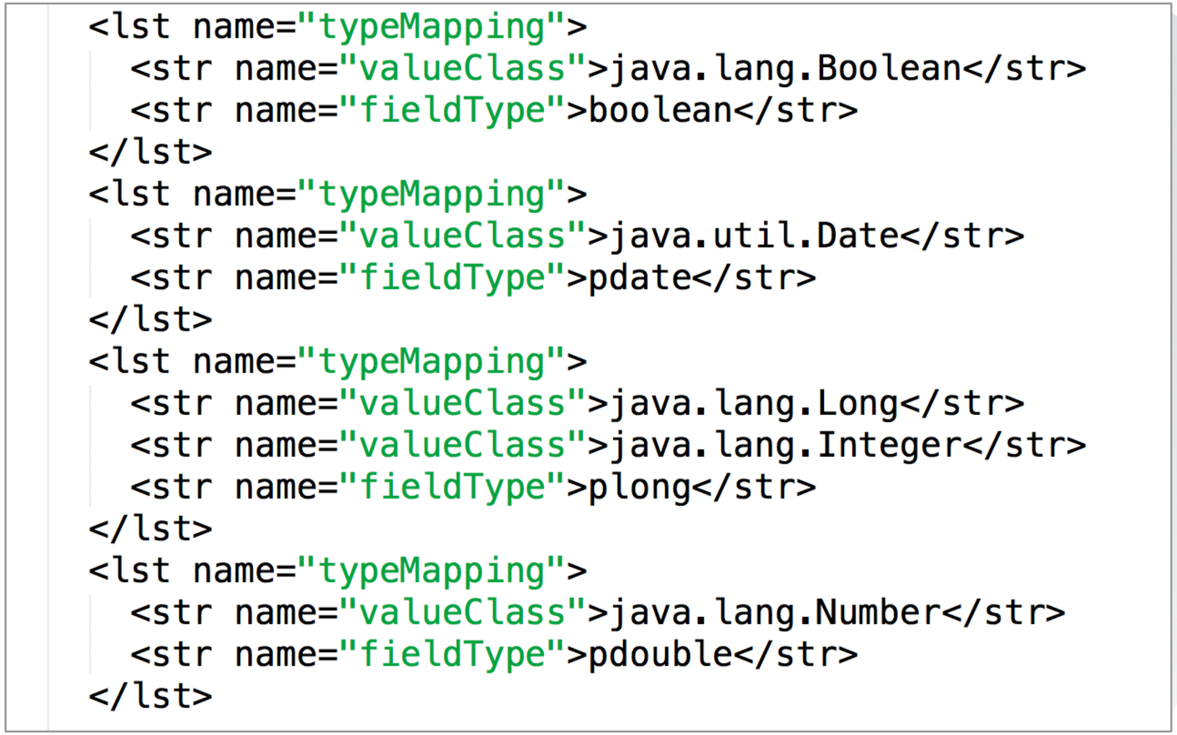
```
    <lst name="typeMapping">
      <str name="valueClass">java.lang.Boolean</str>
      <str name="fieldType">boolean</str>
    </lst>
    <lst name="typeMapping">
      <str name="valueClass">java.util.Date</str>
      <str name="fieldType">tdate</str>
    </lst>
    <lst name="typeMapping">
      <str name="valueClass">java.lang.Long</str>
      <str name="valueClass">java.lang.Integer</str>
      <str name="fieldType">tlong</str>
    </lst>
    <lst name="typeMapping">
      <str name="valueClass">java.lang.Number</str>
      <str name="fieldType">tdouble</str>
    </lst>

    <lst name="regexMapping">
      <str name="regexPattern">^(https?:\/\/[^\s]+(\.(?i)(jpg|png|gif|bmp))$)</str>
      <str name="fieldType">image</str>
    </lst>
    <lst name="regexMapping">
      <str name="regexPattern">^(https?:\/\/\S+$)</str>
      <str name="fieldType">link</str>
    </lst>
    <lst name="regexMapping">
      <str name="regexPattern">(null)</str>
      <str name="fieldType">null</str>
    </lst>
  <lst name="typeTree">
     <lst name="text">
        <lst name="tdouble">
           <lst name="tlong">
              <lst name="tint" />
           </lst>
        </lst>
        <lst name="pdate" />
        <lst name="tdate" />
        <lst name="link">
           <lst name="image" />
        </lst>
        <lst name="boolean" />
     </lst>
  </lst>
```

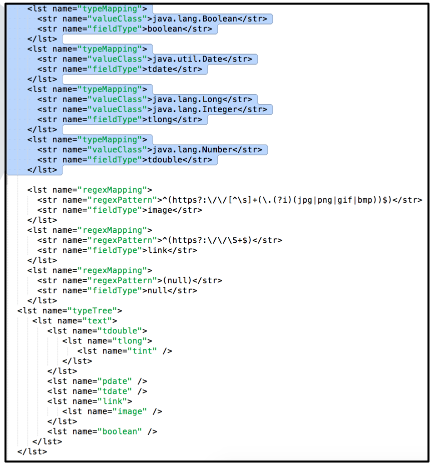
UNBXD

Schema Learning Framework:

Primitives+Regex based inference : Configuring Primitive Types Mapping



```
<lst name="typeMapping">
  <str name="valueClass">java.lang.Boolean</str>
  <str name="fieldType">boolean</str>
</lst>
<lst name="typeMapping">
  <str name="valueClass">java.util.Date</str>
  <str name="fieldType">pdate</str>
</lst>
<lst name="typeMapping">
  <str name="valueClass">java.lang.Long</str>
  <str name="valueClass">java.lang.Integer</str>
  <str name="fieldType">plong</str>
</lst>
<lst name="typeMapping">
  <str name="valueClass">java.lang.Number</str>
  <str name="fieldType">pdouble</str>
</lst>
```



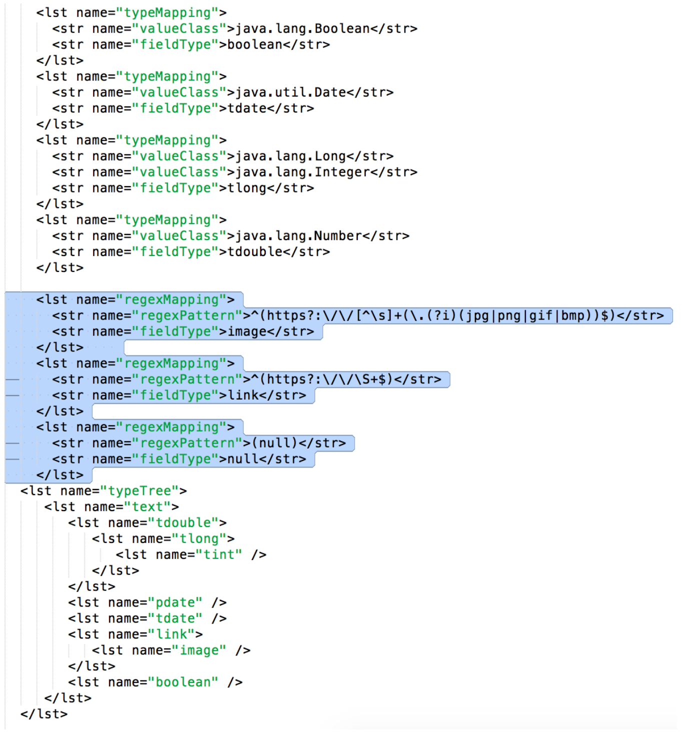
```
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Boolean</str>
    <str name="fieldType">boolean</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.util.Date</str>
    <str name="fieldType">tdate</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Long</str>
    <str name="valueClass">java.lang.Integer</str>
    <str name="fieldType">tlong</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Number</str>
    <str name="fieldType">tdouble</str>
  </lst>

  <lst name="regexMapping">
    <str name="regexPattern">^(https?:\/\/[^\s]+(\.(?i)(jpg|png|gif|bmp))$)</str>
    <str name="fieldType">image</str>
  </lst>
  <lst name="regexMapping">
    <str name="regexPattern">^(https?:\/\/\S+$)</str>
    <str name="fieldType">link</str>
  </lst>
  <lst name="regexMapping">
    <str name="regexPattern">(null)</str>
    <str name="fieldType">null</str>
  </lst>
<lst name="typeTree">
   <lst name="text">
      <lst name="tdouble">
         <lst name="tlong">
            <lst name="tint" />
         </lst>
      </lst>
      <lst name="pdate" />
      <lst name="tdate" />
      <lst name="link">
         <lst name="image" />
      </lst>
      <lst name="boolean" />
   </lst>
</lst>
```

UNBXD

Schema Learning Framework:

Primitives+Regex based inference : Configuring Regex Types Mappings



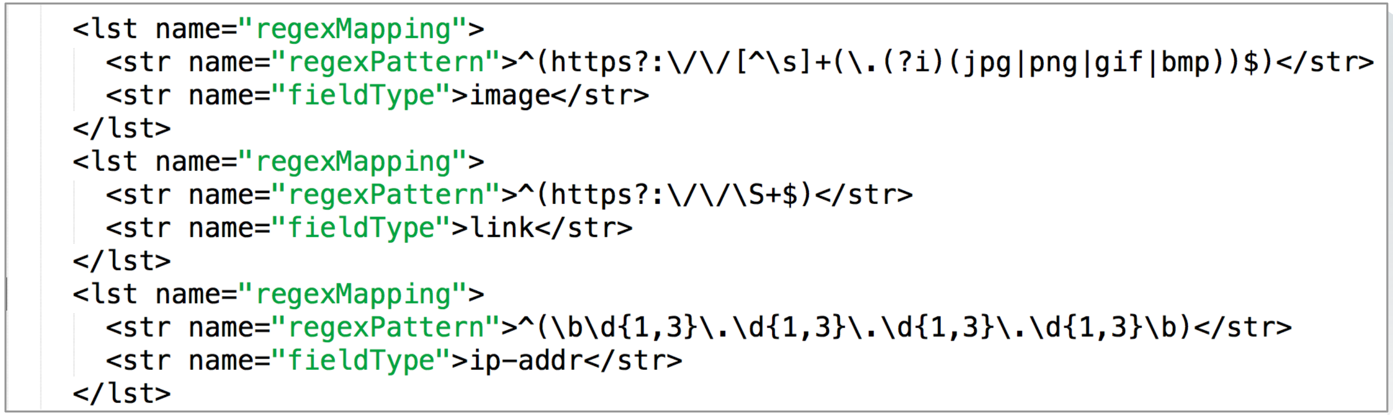
```
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Boolean</str>
    <str name="fieldType">boolean</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.util.Date</str>
    <str name="fieldType">tdate</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Long</str>
    <str name="valueClass">java.lang.Integer</str>
    <str name="fieldType">tlong</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Number</str>
    <str name="fieldType">tdouble</str>
  </lst>

  <lst name="regexMapping">
    <str name="regexPattern">^(https?:\/\/[^\s]+(\.(?i)(jpg|png|gif|bmp))$)</str>
    <str name="fieldType">image</str>
  </lst>
  <lst name="regexMapping">
    <str name="regexPattern">^(https?:\/\/\S+$)</str>
    <str name="fieldType">link</str>
  </lst>
  <lst name="regexMapping">
    <str name="regexPattern">(null)</str>
    <str name="fieldType">null</str>
  </lst>
<lst name="typeTree">
   <lst name="text">
      <lst name="tdouble">
         <lst name="tlong">
            <lst name="tint" />
         </lst>
      </lst>
      <lst name="pdate" />
      <lst name="tdate" />
      <lst name="link">
         <lst name="image" />
      </lst>
      <lst name="boolean" />
   </lst>
</lst>
```

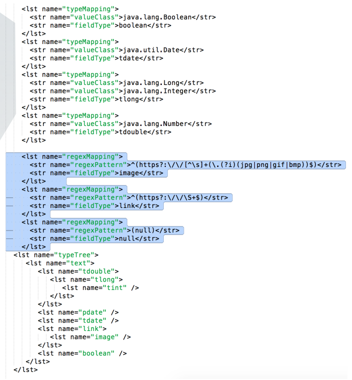
UNBXD

Schema Learning Framework:

Primitives+Regex based inference : Configuring Regex Types Mappings



```
<lst name="regexMapping">
  <str name="regexPattern">^(https?:\/\/[^\s]+(\.(?i)(jpg|png|gif|bmp))$)</str>
  <str name="fieldType">image</str>
</lst>
<lst name="regexMapping">
  <str name="regexPattern">^(https?:\/\/\S+$)</str>
  <str name="fieldType">link</str>
</lst>
<lst name="regexMapping">
  <str name="regexPattern">^(\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b)</str>
  <str name="fieldType">ip-addr</str>
</lst>
```

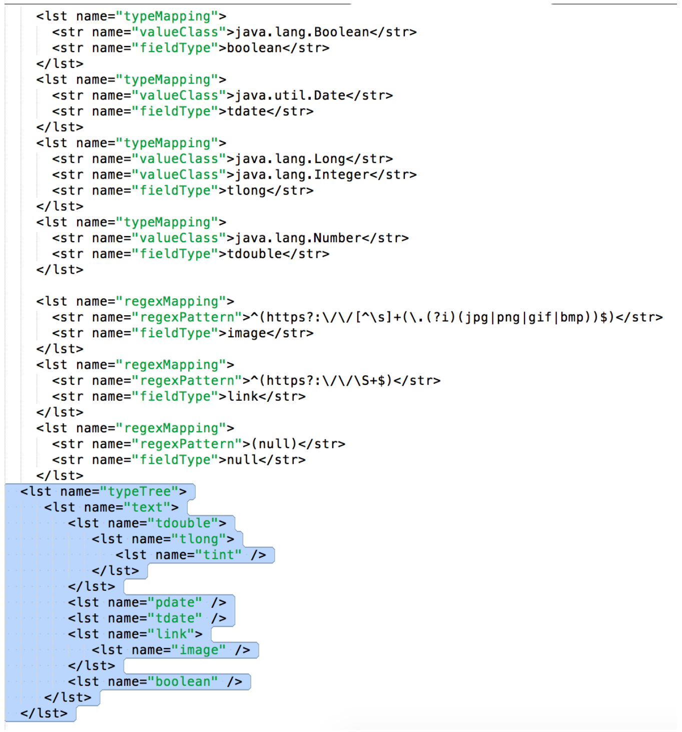


```
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Boolean</str>
    <str name="fieldType">boolean</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.util.Date</str>
    <str name="fieldType">tdate</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Long</str>
    <str name="valueClass">java.lang.Integer</str>
    <str name="fieldType">tlong</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Number</str>
    <str name="fieldType">tdouble</str>
  </lst>

  <lst name="regexMapping">
    <str name="regexPattern">^(https?:\/\/[^\s]+(\.(?i)(jpg|png|gif|bmp))$)</str>
    <str name="fieldType">image</str>
  </lst>
  <lst name="regexMapping">
    <str name="regexPattern">^(https?:\/\/\S+$)</str>
    <str name="fieldType">link</str>
  </lst>
  <lst name="regexMapping">
    <str name="regexPattern">(null)</str>
    <str name="fieldType">null</str>
  </lst>
<lst name="typeTree">
   <lst name="text">
      <lst name="tdouble">
         <lst name="tlong">
            <lst name="tint" />
         </lst>
      </lst>
      <lst name="pdate" />
      <lst name="tdate" />
      <lst name="link">
         <lst name="image" />
      </lst>
      <lst name="boolean" />
   </lst>
</lst>
```

Schema Learning Framework:

Primitives+Regex based inference : Configuring Field-Type Hierarchies



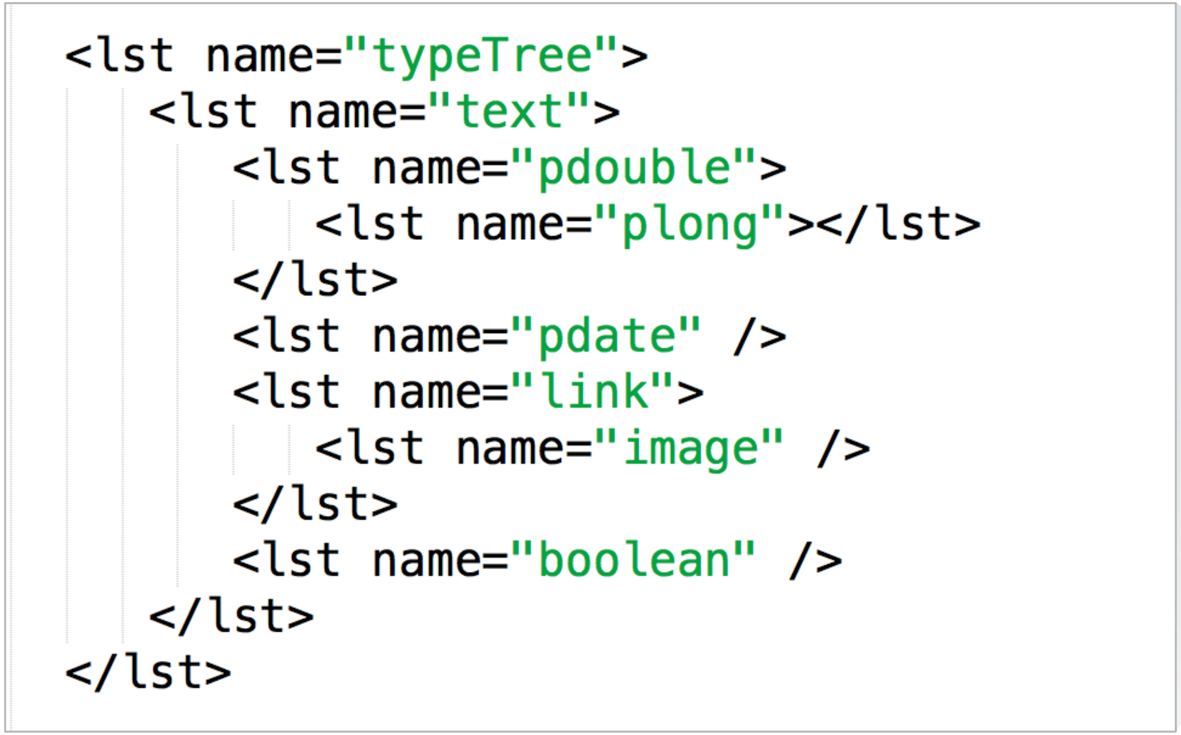
```
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Boolean</str>
    <str name="fieldType">boolean</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.util.Date</str>
    <str name="fieldType">tdate</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Long</str>
    <str name="valueClass">java.lang.Integer</str>
    <str name="fieldType">tlong</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Number</str>
    <str name="fieldType">tdouble</str>
  </lst>

  <lst name="regexMapping">
    <str name="regexPattern">^(https?:\/\/[^\s]+(\.(?i)(jpg|png|gif|bmp))$)</str>
    <str name="fieldType">image</str>
  </lst>
  <lst name="regexMapping">
    <str name="regexPattern">^(https?:\/\/\S+$)</str>
    <str name="fieldType">link</str>
  </lst>
  <lst name="regexMapping">
    <str name="regexPattern">(null)</str>
    <str name="fieldType">null</str>
  </lst>
<lst name="typeTree">
   <lst name="text">
      <lst name="tdouble">
         <lst name="tlong">
            <lst name="tint" />
         </lst>
      </lst>
      <lst name="pdate" />
      <lst name="tdate" />
      <lst name="link">
         <lst name="image" />
      </lst>
      <lst name="boolean" />
   </lst>
</lst>
```

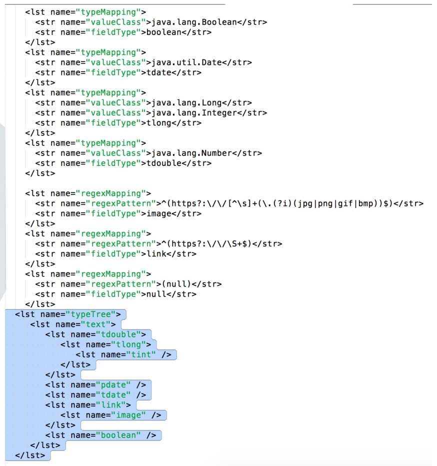
UNBXD

Schema Learning Framework:

Primitives+Regex based inference : Configuring Field-Type Hierarchies



```
<lst name="typeTree">
   <lst name="text">
      <lst name="pdouble">
         <lst name="plong"></lst>
      </lst>
      <lst name="pdate" />
      <lst name="link">
         <lst name="image" />
      </lst>
      <lst name="boolean" />
   </lst>
</lst>
```



```
<lst name="typeMapping">
  <str name="valueClass">java.lang.Boolean</str>
  <str name="fieldType">boolean</str>
</lst>
<lst name="typeMapping">
  <str name="valueClass">java.util.Date</str>
  <str name="fieldType">tdate</str>
</lst>
<lst name="typeMapping">
  <str name="valueClass">java.lang.Long</str>
  <str name="valueClass">java.lang.Integer</str>
  <str name="fieldType">tlong</str>
</lst>
<lst name="typeMapping">
  <str name="valueClass">java.lang.Number</str>
  <str name="fieldType">tdouble</str>
</lst>

<lst name="regexMapping">
  <str name="regexPattern">^(https?:\/\/[^\s]+(\.(?i)(jpg|png|gif|bmp))$)</str>
  <str name="fieldType">image</str>
</lst>
<lst name="regexMapping">
  <str name="regexPattern">^(https?:\/\/\S+$)</str>
  <str name="fieldType">link</str>
</lst>
<lst name="regexMapping">
  <str name="regexPattern">(null)</str>
  <str name="fieldType">null</str>
</lst>
<lst name="typeTree">
   <lst name="text">
      <lst name="tdouble">
         <lst name="tlong">
            <lst name="tint" />
         </lst>
      </lst>
      <lst name="pdate" />
      <lst name="tdate" />
      <lst name="link">
         <lst name="image" />
      </lst>
      <lst name="boolean" />
   </lst>
</lst>
```

UNBXD

Schema Learning Framework:

Primitives based Inference

- Demo

UNBXD

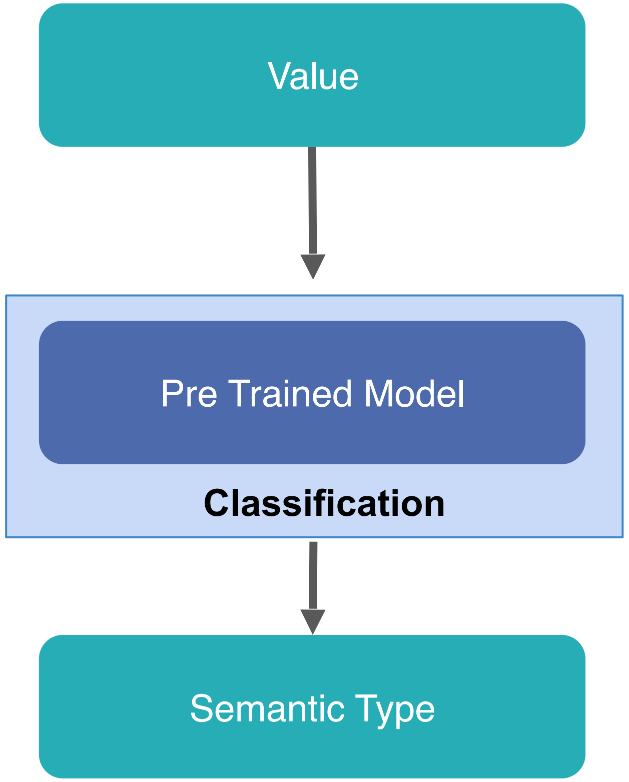
Schema Learning Framework:

Model based inference

- Pluggable Pre-Trained models
- *Semantic Type Inference*

❖ Example:

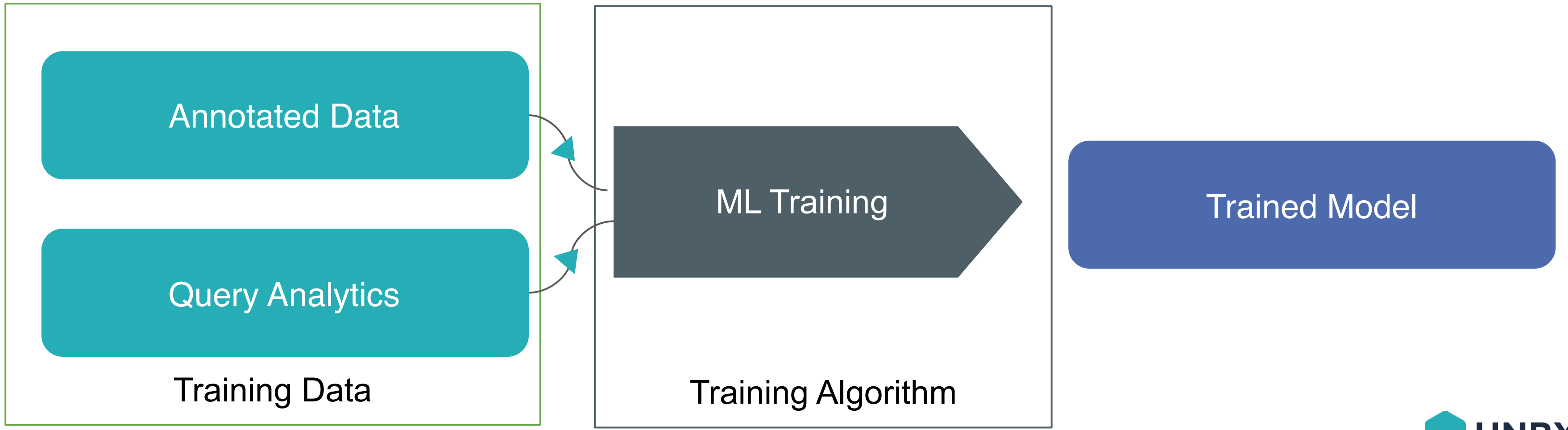
➢ Red	→ Color
➢ Men's Shirt	→ Category
➢ Blue Formal Shirt	→ Title
➢ Gini & Jony	→ Brand



UNBXD

Schema Learning Framework:
Model based inference: Training

- Models can be trained with:
  - Pre-Annotated Judgements : {“grape green”->”color”, “red monsoon shirt ” -> “title” }
  - Query Analysis Data



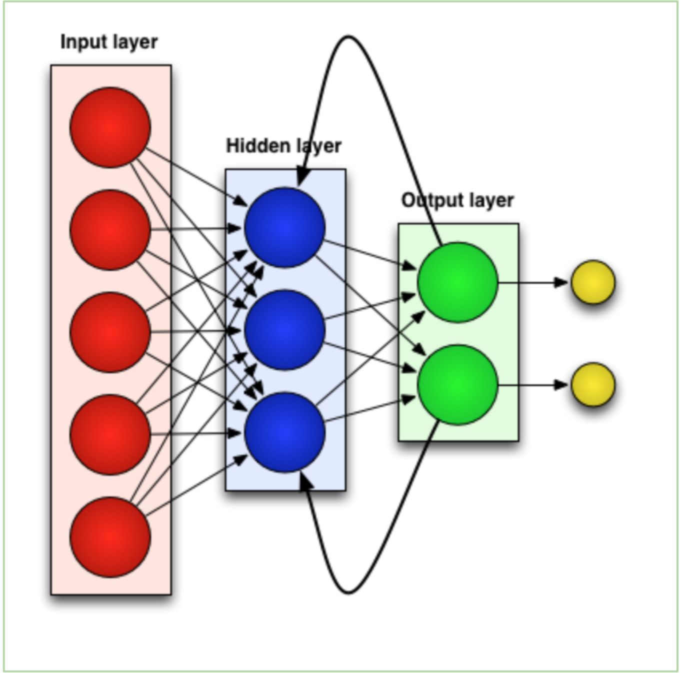
Schema Learning Framework:

Model based inference: Training

- *Recurrent Neural Networks (RNN)*

❖ Feedback loops

❖ Contextual Learning



Schema Learning Framework:

Model based inference

- Demo

UNBXD

Conclusion

- A framework within Solr
- Supports hierarchies of syntactic & semantic types
- Pluggable Domain Specific Inference
- Automated Creation of Accurate Schema

UNBXD

Roadmap

- SOLR-11741, SOLR-6939
  - Chris Hostetter(Hoss Man), Committer Lucene Solr

- Phase 1: Framework, primitives inference & regex inference

- Phase 2: Model inference (pluggability)

UNBXD

The Team

Kishore Angani
Senior Architect, Unbxd

Ishan Chattopadhyaya
Search Architect, Unbxd
Lucene/Solr Committer

Abhishek Kumar Singh
Search Engineer,
Unbxd

Jestin James
Sr. Interaction Designer,
Unbxd

UNBXD