Declarative Text Understanding with SystemT

Huaiyu Zhu, Yunyao Li, Laura Chiticariu, Marina Danilevsky,

Sanjana Sahayaraj, Teruki Tauchi

Tutorial Roadmap

- Introduction to enterprise text understanding
- Overview of SystemT
- Visual introduction to building extractors
- Hands-on Lab 1: Building extractors with Watson Knowledge Studio
- Hands-on Lab 2: Creating and editing extractors visually with the Advanced Rule Editor
- Annotation Query Language (AQL)
- Hands-on Lab 3: Creating advanced extractors and writing AQL with the Advanced Rule Editor
- Advanced topics and research directions

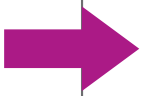
Introduction to Enterprise Text Understanding

Information Extraction (IE)

- Distill structured data from unstructured and semi-structured text
- Enable humans to sift through information faster
- Enable computers to better exploit the text

…
For years, Microsoft Corporation CEO Bill Gates was against open source. But today he appears to have changed his mind. "We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access."

Richard Stallman, founder of the Free Software Foundation, countered saying…



Name	Title	Organization
Bill Gates	CEO	Microsoft
Bill Veghte	VP	Microsoft
Richard Stallman	Founder	Free Soft..

(from Cohen's IE tutorial, 2003)

Two Case Studies

1. Large Retail Bank spends $X billion on an advertising campaign. How effective was it ?
2. Large IT Infrastructure Company is penalized when they violate their service agreement. What hardware faults are responsible ?

Case Study 1: Sentiment Analysis

Social Media

Text Analytics

Product catalog, Customer Master Data, …

Personal Attributes

Relationships

Life Events

Products Interests

360º Profile

Statistical Analysis, Report Gen.

Complexity

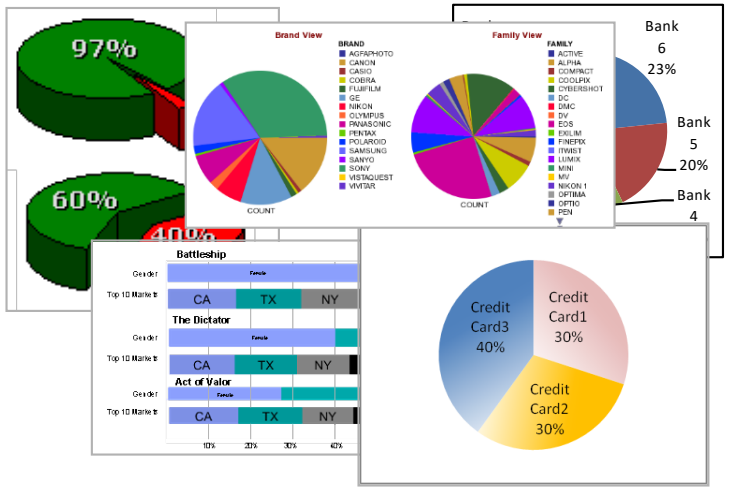
Sarcasm, wishful thinking, …

Breadth

Buzz, intent, sentiment, life events, personal atts, …

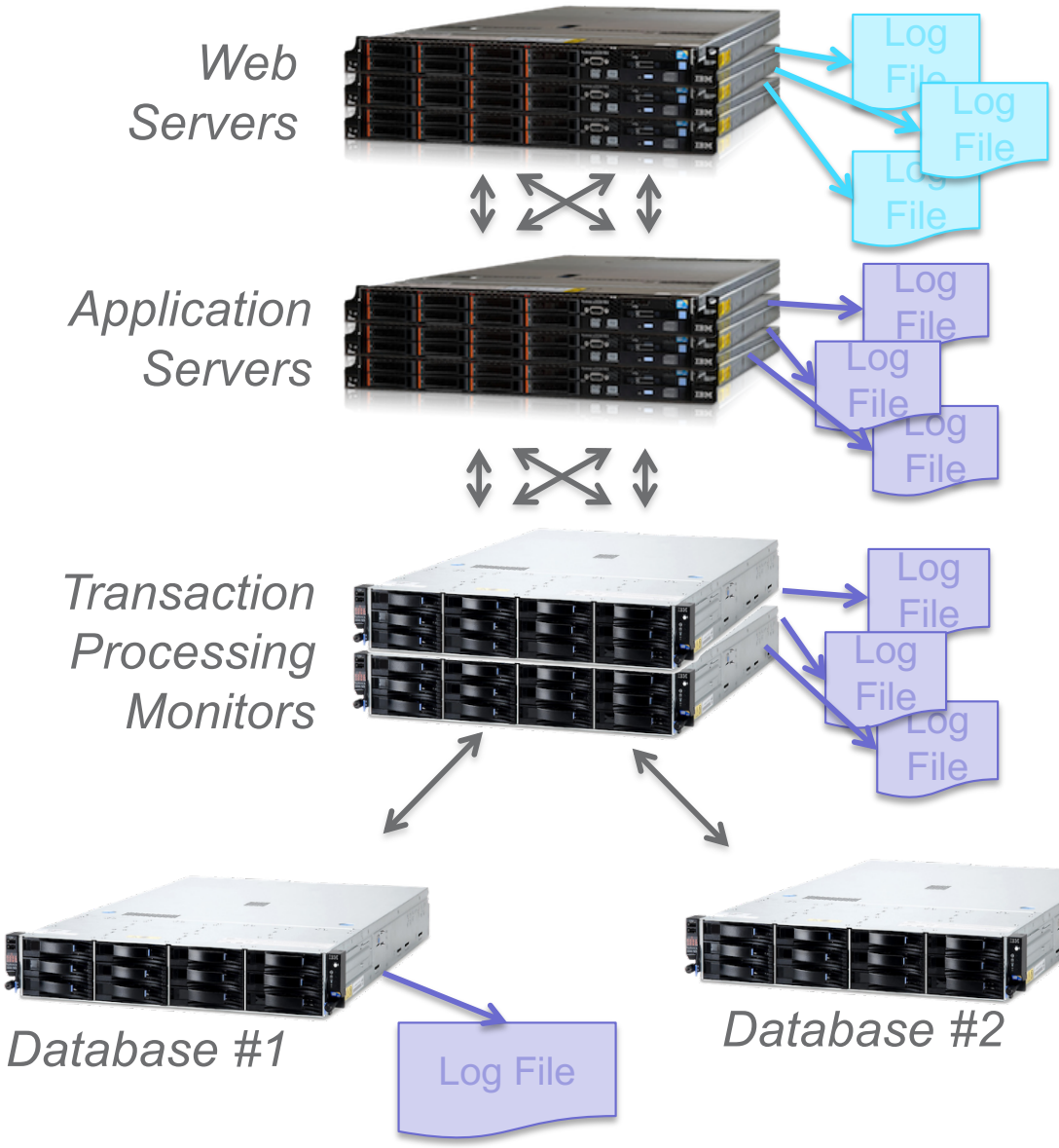
Scale

500M+ tweets a day, 100M+ consumers, …



Customer 360º

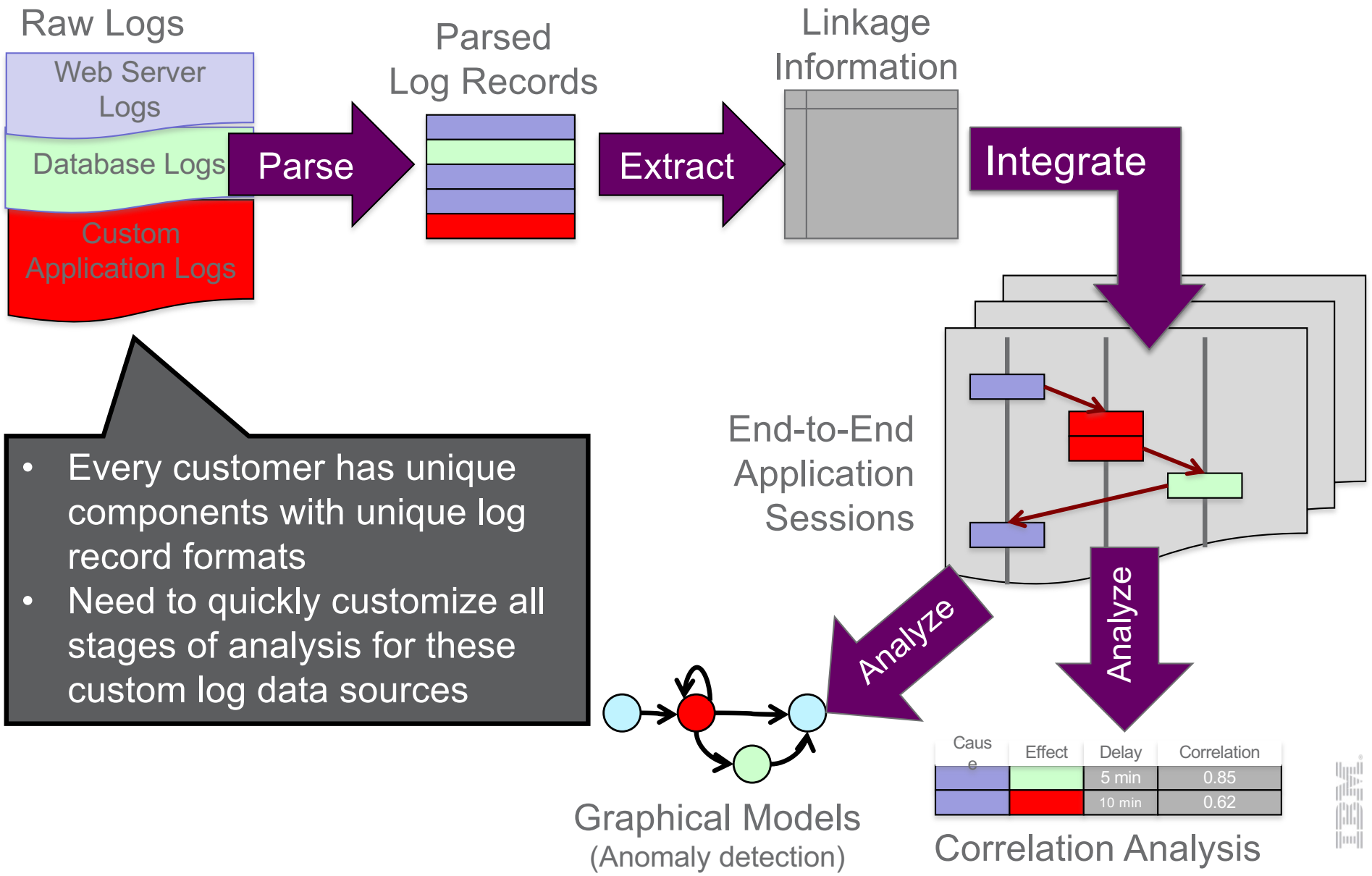
Case Study 2: Machine Analysis



- Web site with multi-tier architecture
- Every component produces its own system logs
- An error shows up in the log for Database #2
- **What sequence of events led to this error?**

```
12:34:56 SQL ERROR 43251:
Table CUST.ORDERWZ is not
```

Case Study 2: Machine Analysis



- Every customer has unique components with unique log record formats
- Need to quickly customize all stages of analysis for these custom log data sources

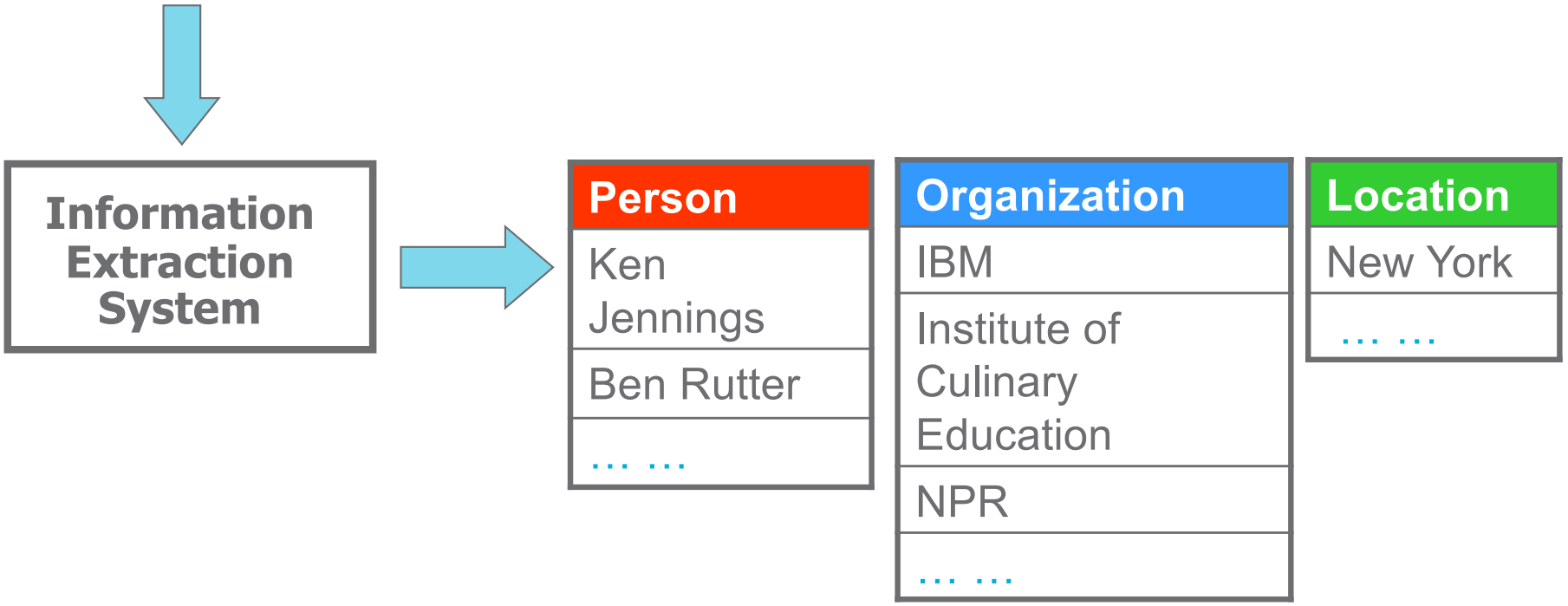
Common across use cases: Text Understanding

- **Information Extraction:** the first step of Text Understanding
  - The information required to find the solution is hidden in the text

- Text Understanding tasks include:
  - Named Entity Recognition (NER)
  - Relation Extraction
  - Event Extraction
  - Sentiment Extraction
  - Co-reference Resolution
  - Table Extraction

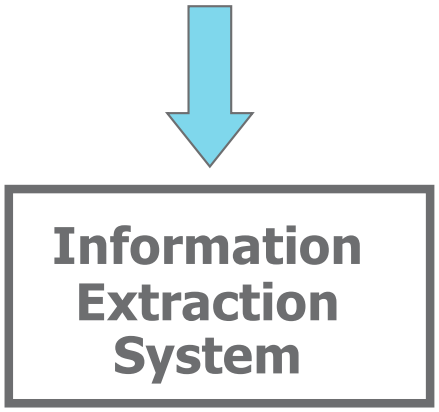
Text Understanding: Named Entity Recognition

First, *Jeopardy!* champions Ken Jennings and Ben Rutter got served by Watson, the IBM supercomputer. Now, Watson is serving up dishes from a food truck as part of a new partnership with the Institute of Culinary Education in New York, NPR reports…



Text Understanding: Relation Extraction

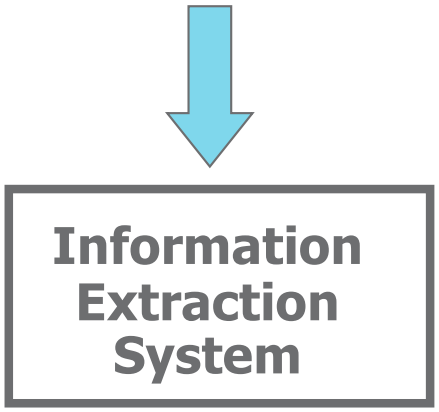
Facebook, based in Menlo Park, Calif., will pay $4 billion in cash and $12 billion worth of shares for WhatsApp. But the ultimate cost of the deal could rise to $19 billion, …



Company 1	Company 2	Cost
Facebook	WhatsApp	$19 billion
…	…	…

Text Understanding: Event Extraction

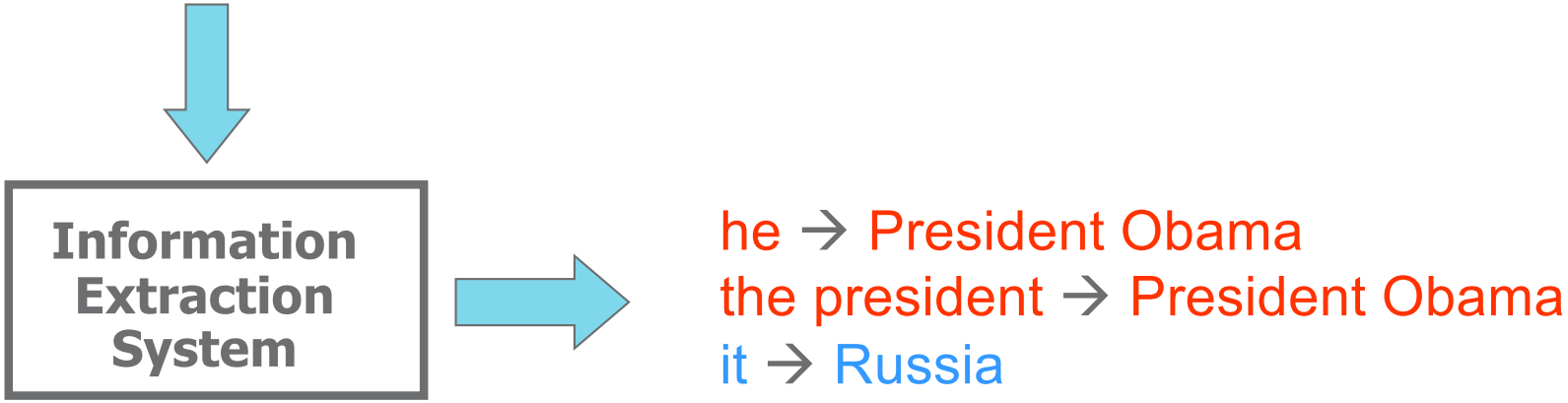
MELBOURNE, Australia -- Three thoughts after Li Na beat Dominika Cibulkova, 7-6 (3), 6-0 to win the 2014 Australian Open women's title…



Winner	Opponent	Year	Event
Li Na	Dominika Cibulkova	2014	Australian Open
…	…	…	…

Text Understanding: Coreference Resolution

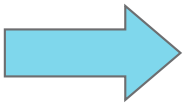
President Obama on Thursday declared that a referendum asking Crimean voters if they want to join Russia would violate Ukraine's constitution and international law, and he promised to keep ramping up the pressure on Russia until it stands down in the region. … , the president said in the White House briefing room.



Text Understanding: Sentiment Analysis

"Hunger Games" is certainly the best film I have seen since many many months, and it is a very successful adaptation of an excellent book.

Information Extraction System



Sentiment	Sentiment Phrase
Positive	“Hunger Games” is certainly the best film
Positive	it is a very successful adaptation of an …
…	…

Text Understanding: Table Extraction

LENDERS' COMMITMENTS

The Charles Schwab Corporation $800,000,000 Credit Agreement (364-Day Commitment) dated as of June 12, 2009.

		Lender Commitment Amount		
1.	Citibank, N.A.	1.	$	90,000,000
2.	JPMorgan Chase Bank, N.A.	2.	$	90,000,000
3.	Bank of America, N.A.	3.	$	80,000,000
4.	PNC Bank, National Association	4.	$	80,000,000
5.	Wells Fargo Bank, National Association	5.	$	80,000,000
6.	Credit Suisse, Cayman Islands Branch	6.	$	80,000,000
7.	The Bank of New York Mellon	7.	$	60,000,000
8.	Calyon New York Branch	8.	$	60,000,000
9.	State Street Bank and Trust Company	9.	$	60,000,000
10.	UBS Loan Finance LLC	10.	$	60,000,000
11.	Comerica Bank	11.	$	30,000,000
12.	Lloyds TSB Bank plc	12.	$	30,000,000
		Total	$	800,000,000

Id	Company	Role	Commitment
1	Charles Schwab Corporation	Borrower	
1	Citibank, N.A.	Lender	$90,000,000
1	JPMorgan Chase Bank, N.A.	Lender	$90,000,000
1	Bank of America, N.A.	Lender	$80,000,000
…	…	…	…

Enterprise Requirements for Information Extraction

- **Accuracy**: Quality of answers is very important!
- **Usability**: The system should be easy to use/program
- **Scalability**: The system should be fast and require few compute resources
- **Expressivity**: "Complex" programs are required to capture nuances of human-readable data
- **Transparency**: The system should be easy to understand and to adapt to a different domain

Overview of SystemT

SystemT Overview

Development Environment

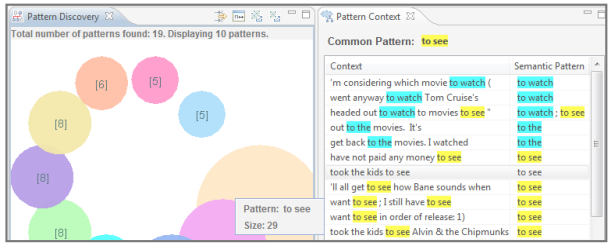
Declarative AQL language

AQL Extractor

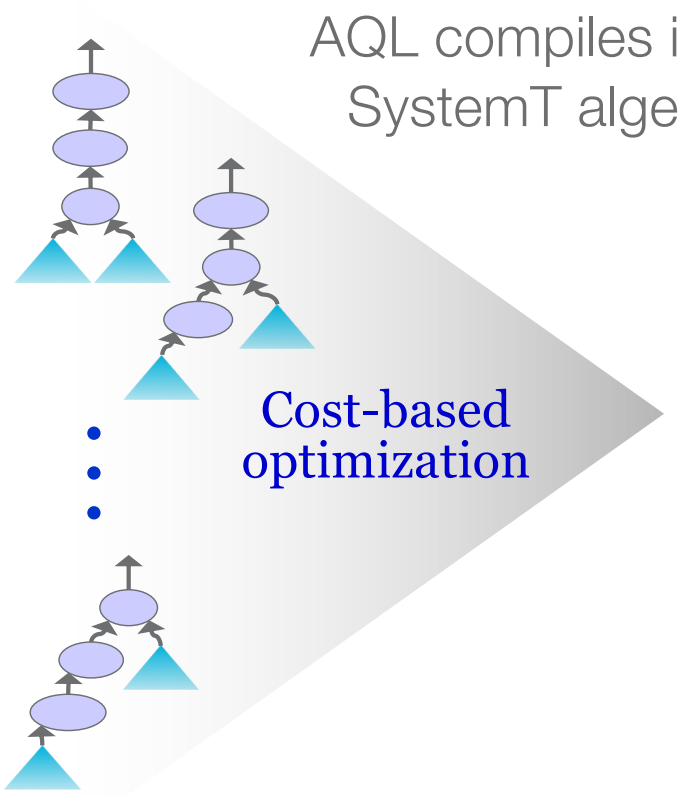
```
create view ProductMention as
  select ...
  from ...
  where ...

create view IntentToBuy as
  select ...
  from ...
  where ...
```

Discovery tools for AQL development



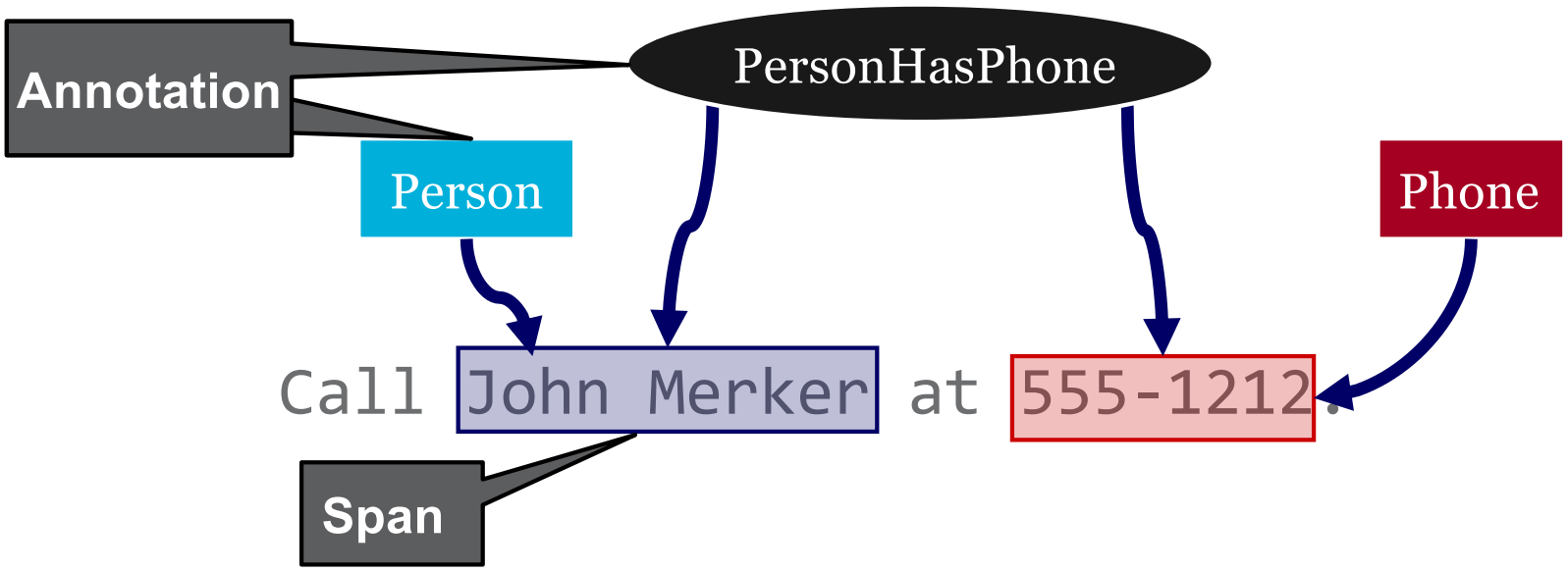
AQL compiles into SystemT algebra



Usability: Separates semantics from implementation
Expressivity: Richer, cleaner rule semantics
Scalability: Better performance through optimization

SystemT Basic Data Model: Annotations and Spans

- Common IE terminology:
  - Each unit of structured information extracted from a document is called an **annotation**
    - Includes both the final outputs of extraction and the lower-level intermediate outputs
  - Annotations are usually associated with one or more regions of text called **spans**



SystemT Basic Data Model: Annotations and Spans

Document
text: STRING

Annotation (Span)		
begin: INT	*end*: INT	*doc*: DOC

- Relational-like model:
  - data is organized in *tuples*
  - tuples have a *schema*
- Special constructs necessary for text processing:
  - Document consists of a single ***text*** attribute
  - Annotations are represented by a type called **Span**,
    - Span consists of ***begin***, ***end*** and ***document*** attributes

“Document at a time” execution model

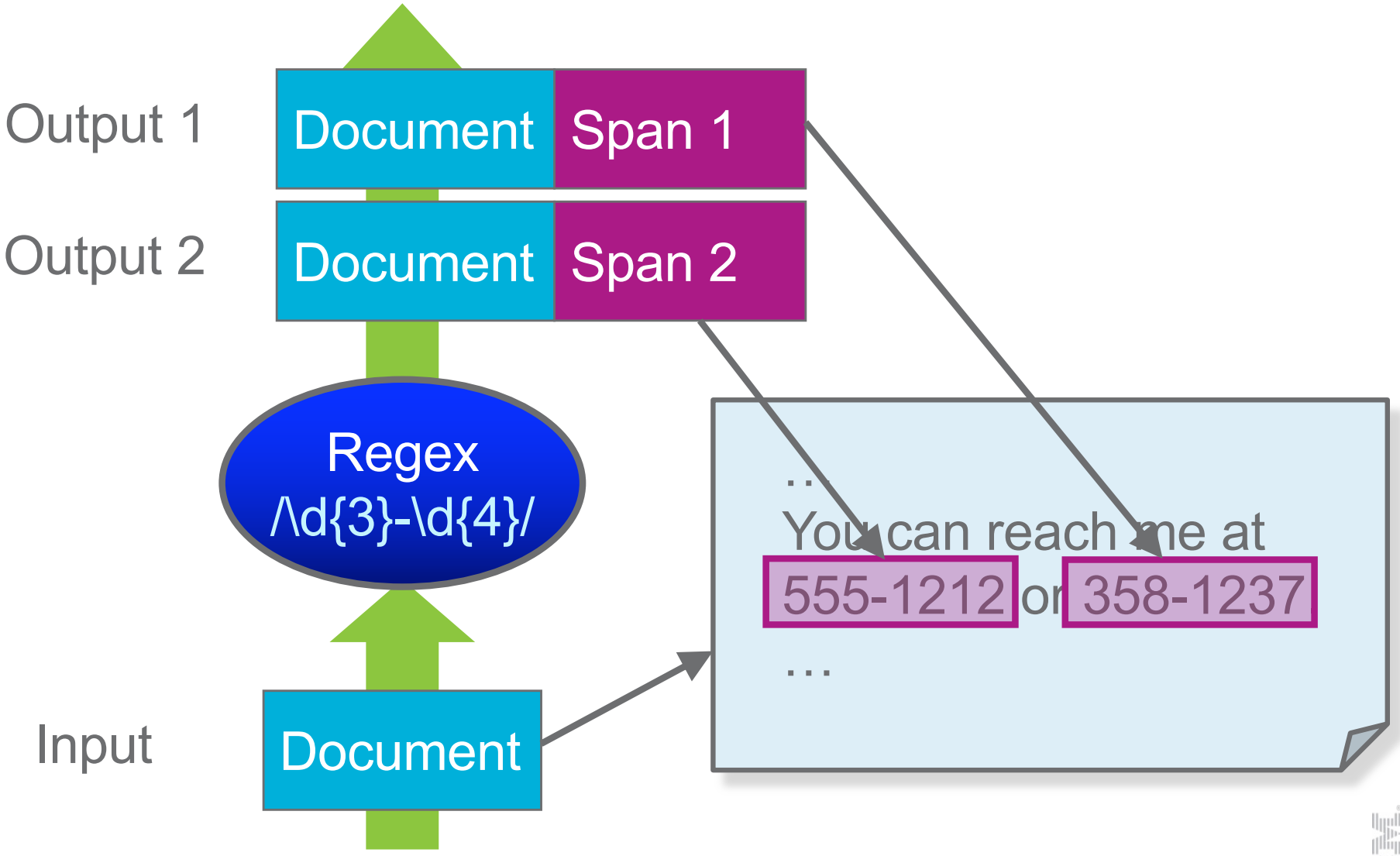
Yahoo Has Flat Sales and Profit, but Shares Surge The rise is because Yahoo shareholders are far more excited about what the company owns — namely Alibaba — than anything the company runs.

Intel Posts 5 Percent Lower Earnings for First Quarter With PC sales falling, the semiconductor maker reported a drop in year-over-year quarterly earnings, to $1.95 billion

Doc 1	Yahoo
Doc 1	Yahoo
Doc 1	Alibaba
Doc 2	Intel
Doc 2	$1.95 billion

Building Extractors: A Visual Introduction

Example Extractor: Regular Expression



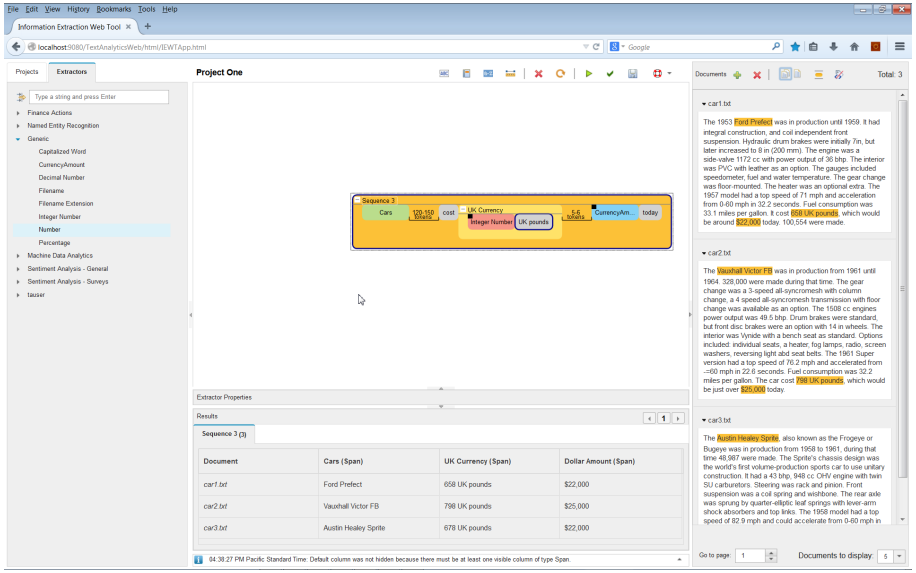
Building Extractors Visually: Two IDEs

Watson Knowledge Studio Rule Editor

- Basic extractor building
- Development by example
- Use dictionaries, regexes, tokens, parts of speech
- Default output refinement
- Available on IBM cloud

Advanced Rule Editor

- Advanced level extractor building
- Workflow-driven development
- Use dictionaries, regexes, tokens, and rich library of pre-built extractors
- Sophisticated output refinement: combine, filter, consolidate
- Exposes AQL, the language of SystemT
- In closed beta on IBM cloud



Building Extractors - Visual Constructs

Atomic constructs

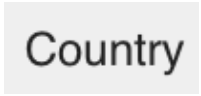
Literal
a single string

Dictionary
list of terms

Regex
regular expression

Proximity
range of tokens or characters

Part of Speech
noun, verb, adjective, …



Pre-built
existing extractor

Composite constructs

Sequence
user-defined sequence of atomic constructs

Union
merged output of two or more extractors

Output refinement

Projection
specific attributes from extractor

Expression
result of applying a scalar function on an attribute

Filter
remove unwanted results from output

Consolidation
handle overlapping matches with user-specified policy

Building Extractors - Visual Constructs

	Watson Knowledge Studio Rule Editor	Advanced Rule Editor
Atomic Constructs		
Literal	✓	✓
Dictionary	✓	✓
Regex	✓	✓
Proximity	✓ *Tokens only*	✓ *Tokens and characters*
Part of speech	✓	✓
Prebuilt		✓
Composite Constructs		
Sequence	✓ *Within sentence only*	✓ *Cross-sentence*
Union	✓	✓
Output Refinement		
Projection		✓
Expression		✓
Filter		✓
Consolidation	✓ *Single policy*	✓ *Multiple policies*

Hands-on Lab 1:
Building Extractors with Watson Knowledge Studio Rule Editor

Hands-on Lab 2:
Creating and Editing Extractors Visually with the Advanced Rule Editor

Annotation Query Language (AQL)

From visual constructs to AQL

```
create view FirstCaps as
select CombineSpans(F.name, C.name) as name
from First F, Caps C
where FollowsTok(F.name, C.name, 0, 2);
```

Optimizer

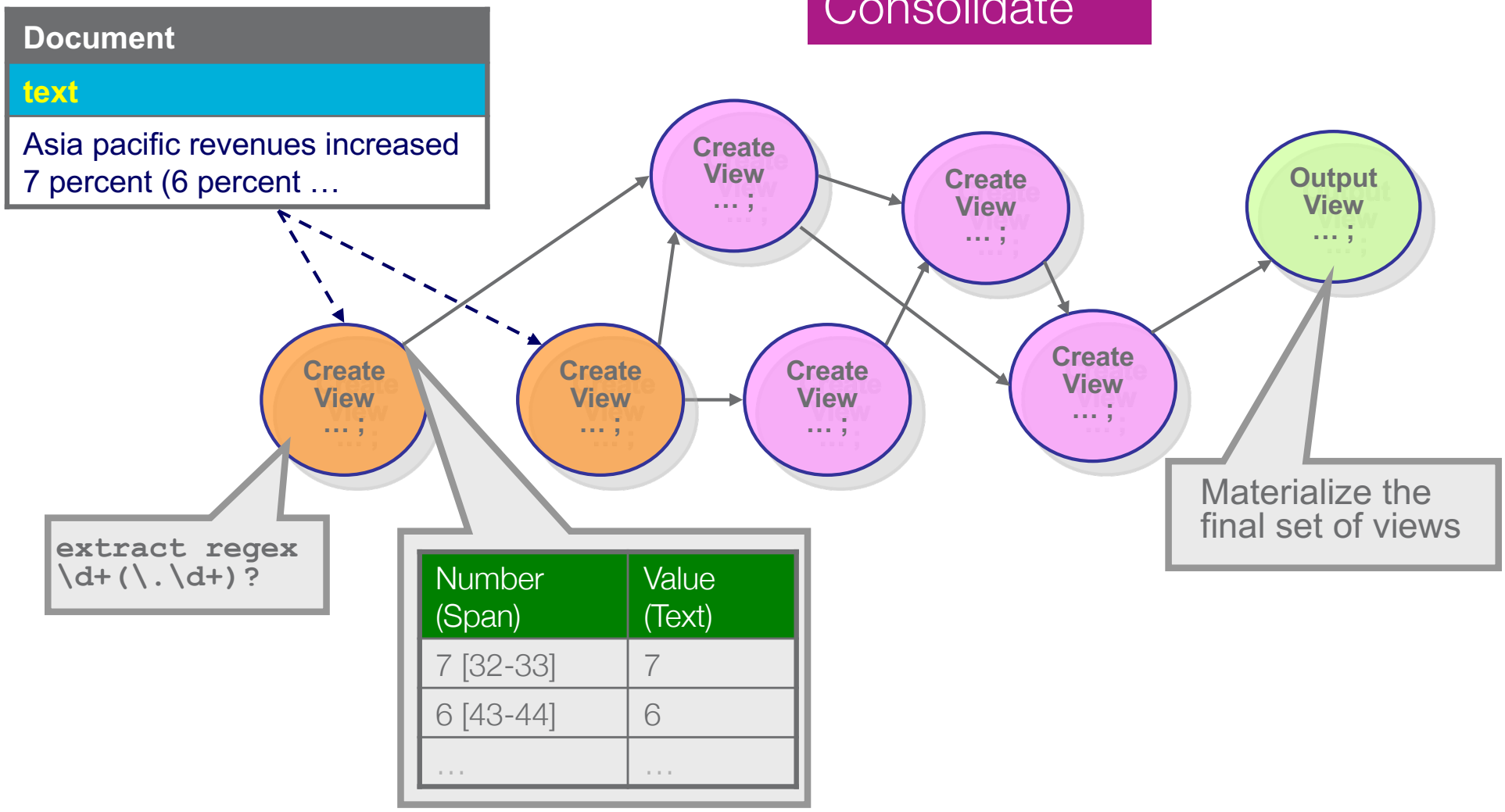
Compiled Plan
(SystemT Algebra)

AQL: Annotation Query Language

- **Data model:** Similar to relational data model
  - A record of an extraction is a **tuple**, similar to a row in a relational table.
  - A tuple is made up of **field**s, each has a **name** and a **type.**
  - A collection of tuples, of the same **schema** (the names and types of all fields of a tuple), can forms a **view**, similar to a relational table.
- **Programming constructs**: Description of creations of data
  - Collection of '**create view**' and '**output view**' statements
  - Constructs used as basic building blocks in the statements:
    - Dictionaries, regular expressions, functions, literal values, …
- **Programming organization**: Facilitating reusability
  - Statements are organized into **modules**, providing separate name spaces for complex programs
  - Names in modules can be **export**ed and **import**ed into other modules.

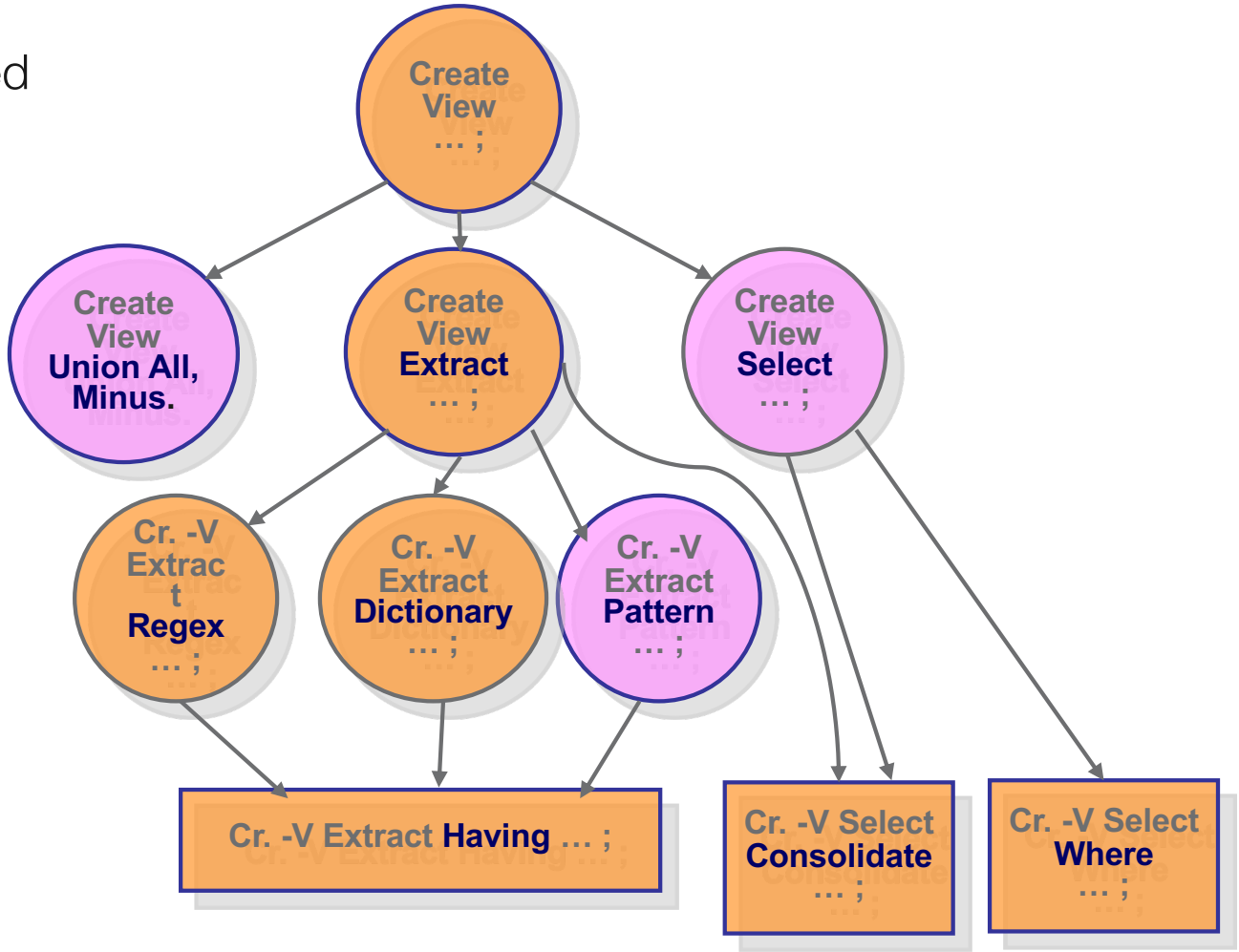
AQL Program

Extract
Select
Filter
Consolidate



The EXTRACT and SELECT Statements

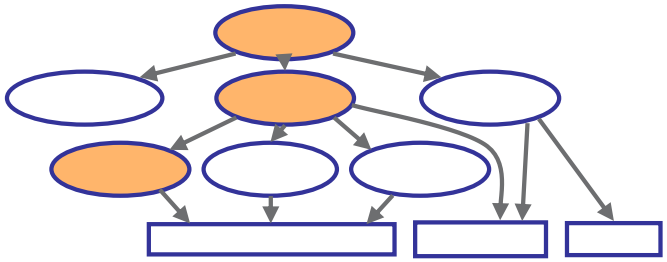
- **Ways to create a view:**
  - Extract new information
  - Process already extracted information
- **Extract** statement – Extraction based on
  - Regular Expressions
  - Dictionaries
  - Sequence Patterns
  - …
  - Key clauses:
    - Having
    - Consolidate
- **Select** statement
  - Key clauses:
    - Where
    - Consolidate
    - …



Full documentation of AQL can be found at https://ibm.biz/aql_reference

Basic EXTRACT Statement: Regular Expressions

Extracting Integers and decimal numbers (without exponents)



```
create view Number as
extract regex
    /\d+(\.\d+)?/
    on D.text as match
from Document D;
```

- Certain information can be easily described as a pattern of characters
- E.g. Telephone numbers, IP address, SSN

Total revenues of $26.3 billion, up 7 percent as reported;

Diluted earnings of $2.26 per share from continuing operations, up 12 percent as reported, or 7 percent compared with the fourth-quarter 2005, excluding pension curtailment charge;

Services signings of $17.8 billion, up 55 percent.

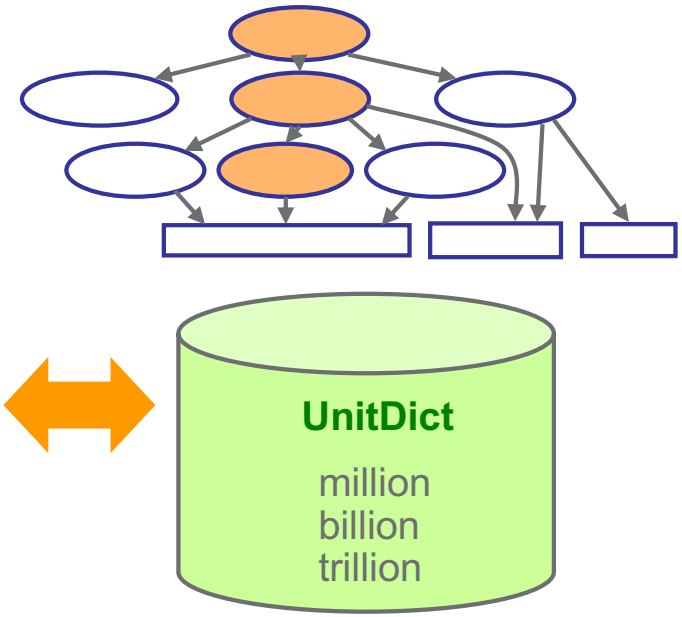
Number
match
26.3
7
2.26
…
55

Basic EXTRACT Statement: Dictionaries

Dictionary for money units

```
-- Creating the dictionary inline
   create dictionary UnitDict
   with language as 'en'
   as ( 'million', 'billion', 'trillion');
```

```
-- Using a file as dictionary
create dictionary UnitDict
from file '<path to your dictionary here>'
with language as 'en';
```

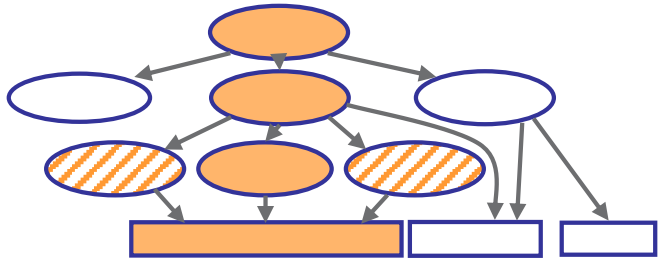



```
-- Using dictionary for extraction
create view Unit as
extract dictionary UnitDict
      with flags 'IgnoreCase'
on D.text as match
from Document D;
```

… Asia-Pacific revenues increased 7 percent (5 percent, adjusting for currency) to $4.8 **billion**. OEM revenues were $1.0 **billion**, down 3 percent compared with the 2005 fourth quarter. …

Basic EXTRACT Statement: The Having clause

```
create view Division as
extract dictionary 'DivisionDict'
      with flags 'IgnoreCase'
on D.text as match
from Document D
having MatchesRegex(/[A-Z].*/, match);
```



…

Revenues from the **Software** segment were $8.6 billion, an increase of 2 percent

…

Revenues from the WebSphere family of **software** products …

Results from dictionary match are filtered by the *expression* in *having* clause

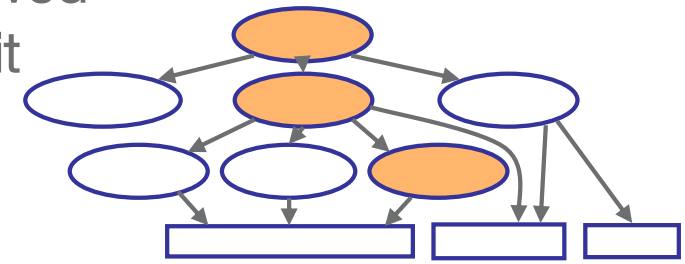
**** having clause can be used with any extract statement***

Basic EXTRACT Statement: Sequence Patterns

Extracting the pattern of a Currency Symbol followed by a number followed optionally by a currency unit

Pattern constructed from

- <View_name.Column_name>
- Regular Expressions
- Tokens
- Strings



```
create view AmountWithUnit as
extract pattern
    '$'<N.match> <U.match>? as match
from Number N, Unit U;
```

Extraction logic specified on previously created views

…

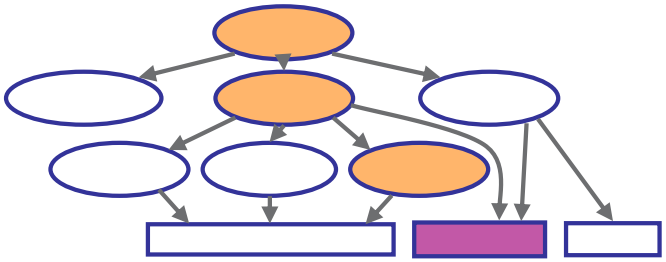
OEM revenues were **$1.0 billion**, down 3 percent

…

AmountWithUnit
match
$1.0
$1.0 billion

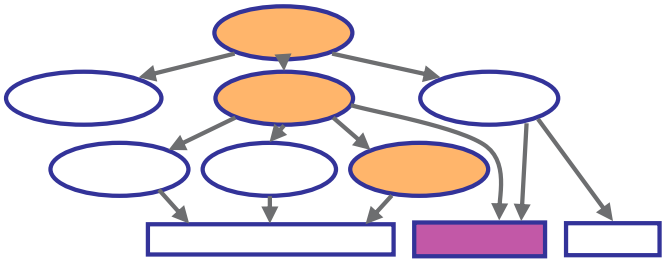
Optional Clauses in EXTRACT PATTERN Statements

```
create view AmountWithUnit as
extract pattern
    '$'<N.match> <U.match>? as match
from Number N, Unit U;
```



Optional Clauses in EXTRACT PATTERN Statements

```
create view AmountWithUnit as
extract pattern
    '$'<N.match> <U.match>? as match
from Number N, Unit U;
```



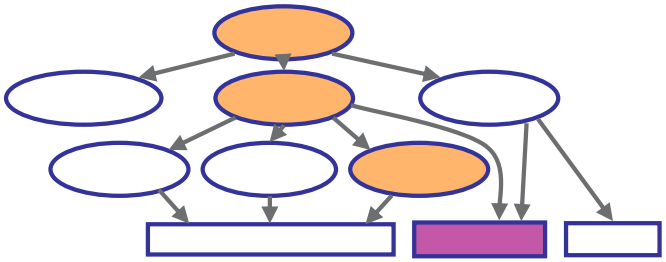
OR

```
create view AmountWithUnit as
extract pattern '$'<N.match> <Token>{0,0}(<U.match>)?
    return group 0 as match
    and group 1 as unit
from Number N, Unit U
consolidate on match using 'ContainedWithin' ;
```

- The group 0 Span is returned by default
- Other groups can be returned

Optional Clauses in EXTRACT PATTERN Statements

```
create view AmountWithUnit as
extract pattern
    '$'<N.match> <U.match>? as match
from Number N, Unit U;
```



OR

Can specify minimum/maximum intervening tokens

```
create view AmountWithUnit as
extract pattern '$'<N.match> <Token>{0,0}(<U.match>)?
    return group 0 as match
    and group 1 as unit
from Number N, Unit U
consolidate on match using 'ContainedWithin' ;
```

- The group 0 Span is returned by default
- Other groups can be returned

Consolidate clause specifies handling of overlapping matches. ContainedWithin is default

Optional Clauses in EXTRACT PATTERN Statements

```
create view AmountWithUnit as
extract pattern '$'<N.match> <Token>{0,0}(<U.match>)?
    return group 0 as match
    and group 1 as unit
from Number N, Unit U
consolidate on match using 'ContainedWithin' ;
```

Total revenues of $26.3 billion, up 7 percent as reported;

Diluted earnings of $2.26 per share from continuing operations, up 12 percent as reported, or 7 percent compared with the fourth-quarter 2005, excluding pension curtailment charge;

Services signings of $17.8 billion, up 55 percent.

AmountWithUnit	
match	**unit**
$26.3 billion	billion
$2.26	<null>
$17.8 billion	billion

Relational Style Statements : SELECT

```
create view AmountWithUnit as
extract pattern
    <N.match> <U.match> as match
from Number N, Unit U;
```

OR

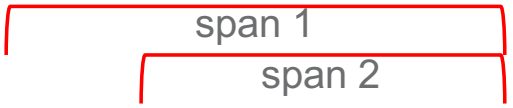
```
create view AmountWithUnit as
select CombineSpans(N.match, U.match) as match
from Number N, Unit U
where FollowsTok(N.match, U.match, 0, 0);
```

The optional where clause:

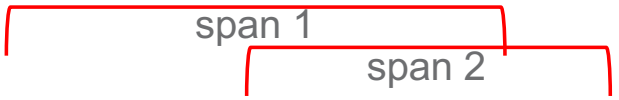
Defines a predicate over the cross product of the input relations of the select statement, as a conjunction of Boolean-returning scalar functions

Relational Style Statements : SELECT

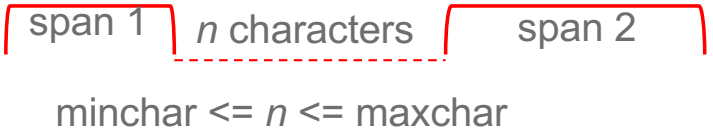
- Predicate functions – Part I:
  - **Contains**(<span1>, <span2>)
    - including touching at end points



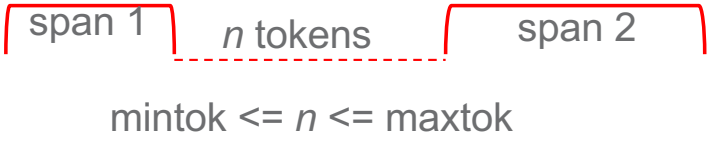
  - **Overlaps**(<span1>, <span2>)
    - not if only touching at end points



  - **Follows**(<span1>, <span2>, <minchar>, <maxchar>)

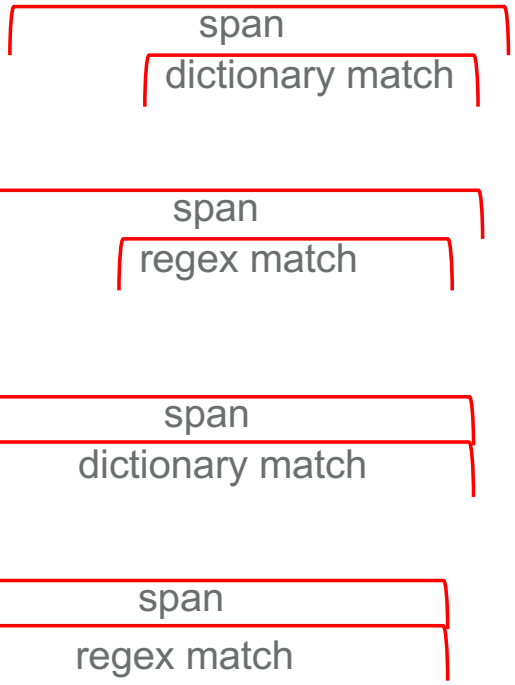


  - **FollowsTok**(<span1>, <span2>, <mintok>, <maxtok>)



Relational Style Statements : SELECT

- Predicate functions – Part II:
  - **ContainsDict**('<dictionary>', ['<flags>',])
  - **ContainsRegex**('<regex>', ['<flags>',])
  - **MatchesDict**('<dictionary>', ['<flags>',])
  - **MatchesRegex**('<regex>', ['<flags>',])

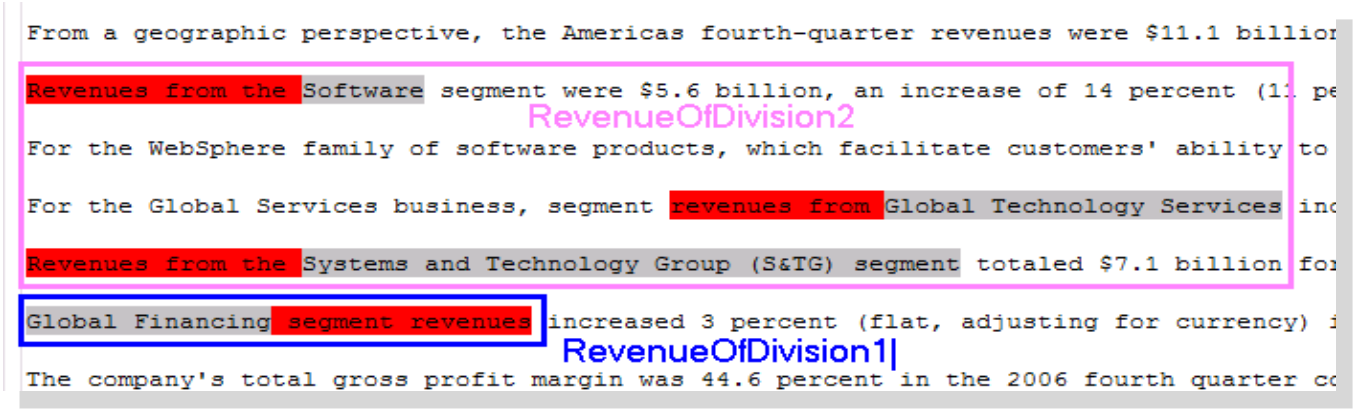


Relational Style Statements : SELECT

- Predicate functions – Part III:
  - **Equals**(<arg1>, <arg2>) — arg1 == arg2
    - Polymorphic type
  - **GreaterThan**(<int1>, <int2>) — int1 > int2
  - **NotNull**(<arg>) — *Is the value of* `arg` *not NULL?*
  - **And**(<boolean_arg1>,<boolean_arg2>, …)
  - **Or**(<boolean_arg1>, <boolean_arg2>)
  - **Not**(<boolean_arg>)

Logical operators

Relational Style statements : UNION ALL




```
create view RevenueOfDivision1 as
select CombineSpans(D.match, R.match)
    as match,
    D.match as division
from Revenue R, Division D
where FollowsTok(D.match, R.match, 0,1);
```

```
create view RevenueOfDivision2 as
extract
  pattern <R.match> ('from'|'for')
          <Token>{0,2} (<D.match>)
return group 0 as match
   and group 2 as division
from Revenue R, Division D;
```

```
create view RevenueOfDivision as
(select R.* from RevenueOfDivision1 R)
union all
(select R.* from RevenueOfDivision2 R);
```

Relational Style statements: MINUS

```
create view NonDollarAmountWithUnit as
(select A.* from AmountWithUnit A)
minus
(select A.* from AmountWithUnit A
 where ContainsRegex(/\$/, LeftContextTok(A.match, 1)));
```

…

Total revenues of **$28.9 billion**, up 10 percent; …

there were **1.51 billion** basic common shares outstanding

…

AmountWithUnit	
match	**unit**
$28.9 billion	**billion**
1.51 billion	**billion**

NonDollarAmountWithUnit	
match	**unit**
1.51 billion	**billion**

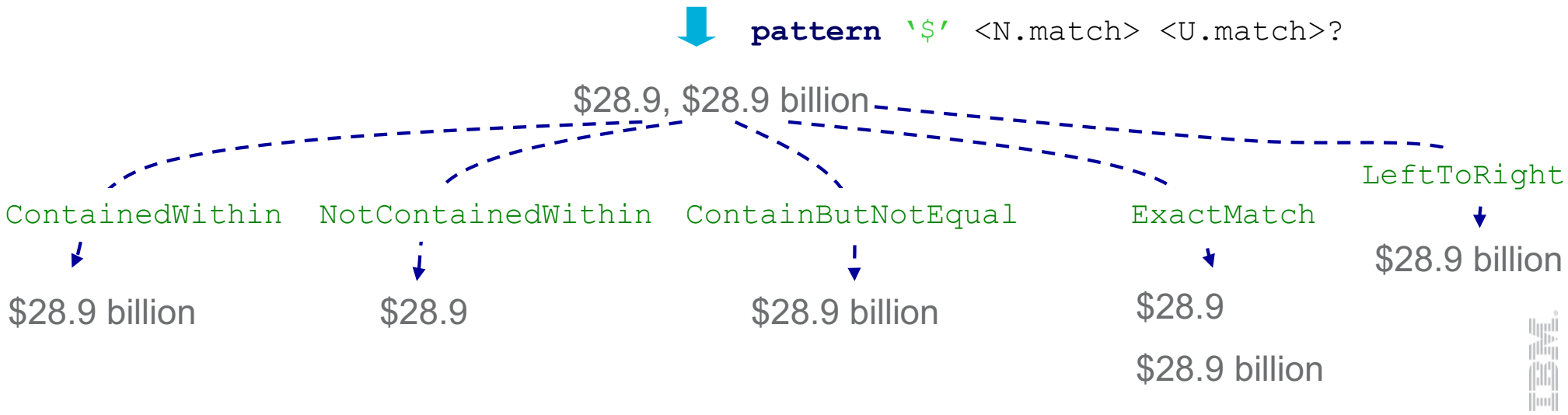
Aggregation Operators: CONSOLIDATE

Optional clause to direct the system what to do if the other parts of the select statements produce spans that overlap

- Consolidate policies
  - **`ContainedWithin`** ➔ Keep the longest match and deduplicate (*default policy*)
  - **`NotContainedWithin`** ➔ Keep the shortest match and deduplicate
  - **`ContainsButNotEqual`** ➔ Keep longest match but keep duplicate matches
  - **`ExactMatch`** ➔ Deduplicate
  - **`LeftToRight`** ➔ Keep the leftmost longest non-overlapping span

… Total revenues of $28.9 billion, up 10 percent; …

Sample input document



The DETAG statement

Task: perform extraction on semi-structured text (HTML, XML)

```
detag <input view name>.<text column> as <output view name>
[detect content_type (always|never)]
[annotate element '<element name>' as <view name>
                  [with attribute '<attribute name>' as <column name>]
                  [and attribute '<attribute name>' as <column name>]
        [, element ...]]
```

```
detag Document.text as DetaggedDoc
detect content_type always
annotate element 'a' as Anchor
 with attribute 'href' as href;
```

Document.text

<html><body>

Today, IBM and Microsoft announced annual results.

</body></html>

Spans over DetaggedDoc.text

Use the `Remap()` scalar function to map them back to original text

Anchor

match	href
IBM	ibm.com
Microsoft	microsoft.com

Hands-on Lab 3:
Creating Advanced Extractors and Writing AQL with the Advanced Rule Editor

Advanced Topics & Research Directions

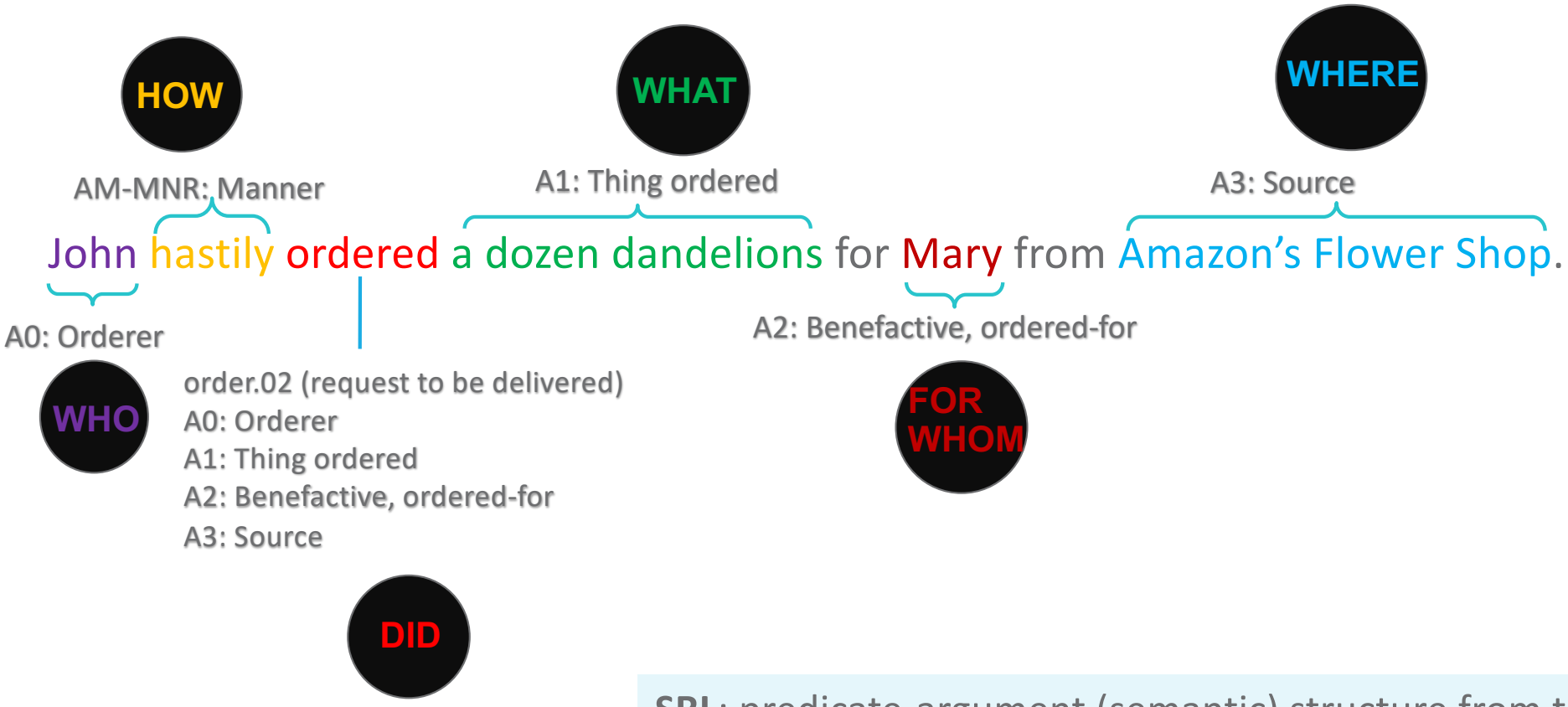
Advanced Topics

- History of Information Extraction systems
  - Historical development of rule-based IE
  - Classical grammar-based IE systems; their expressivity and performance shortcomings
- SystemT's declarative approach to Information Extraction
  - Extending relational algebra with additional text-based operators
  - Addressing the limitations of grammar-based IE systems
  - Optimization in SystemT
- AQL best practices
  - Writing high-performance extraction rules in AQL

SystemT MOOC

- Advanced problems in Text Understanding
  - Shallow semantic parsing aka semantic role labeling
  - Cross-lingual information extraction (in AQL)
  - Table understanding

Shallow Semantic Parsing aka Semantic Role Labeling



SRL: predicate-argument (semantic) structure from text

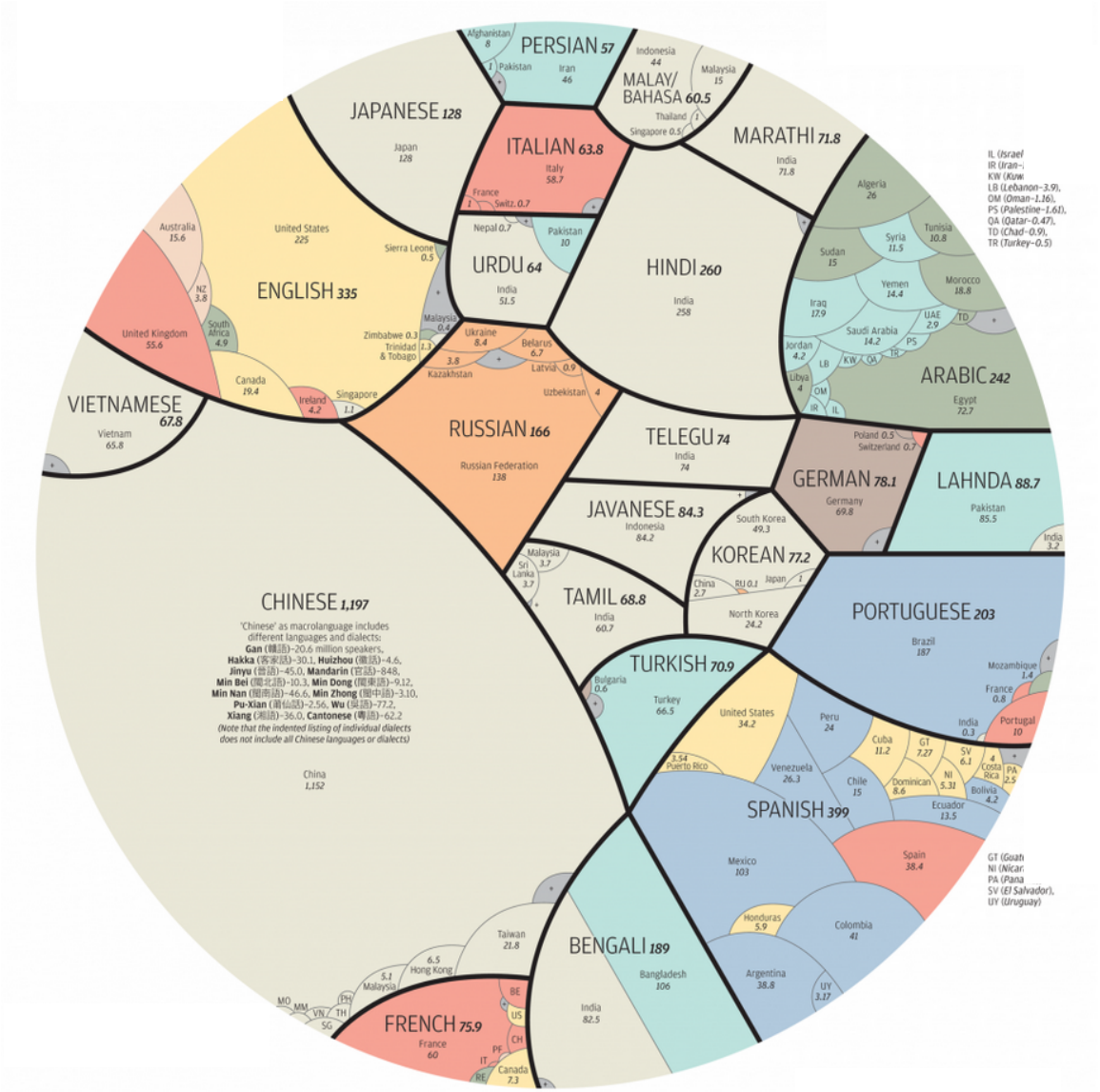
Who did what to whom, when, where and how?

How can we scale out multilingual support?

7,102
known languages

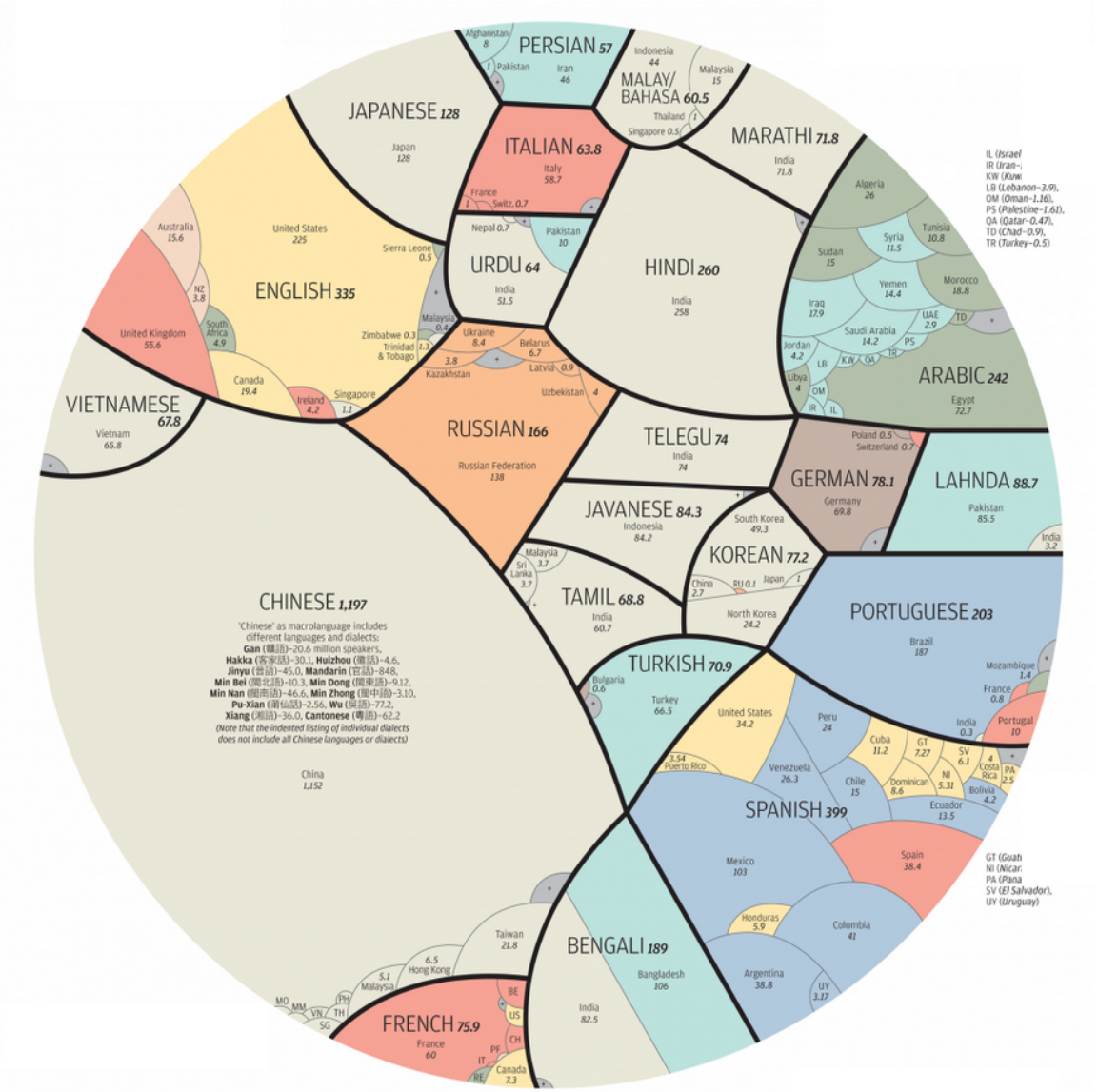
23
most spoken languages

4.1+ Billion
people

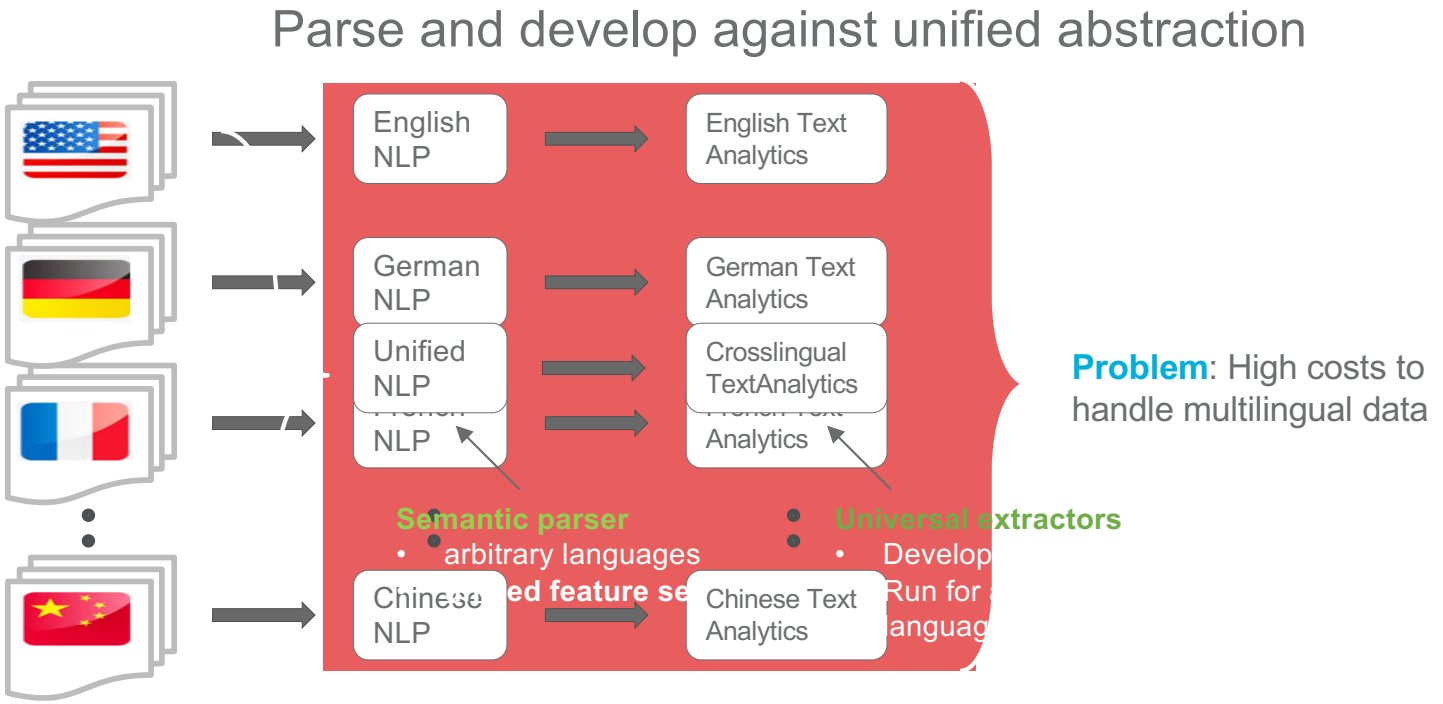


How can we scale out multilingual support?

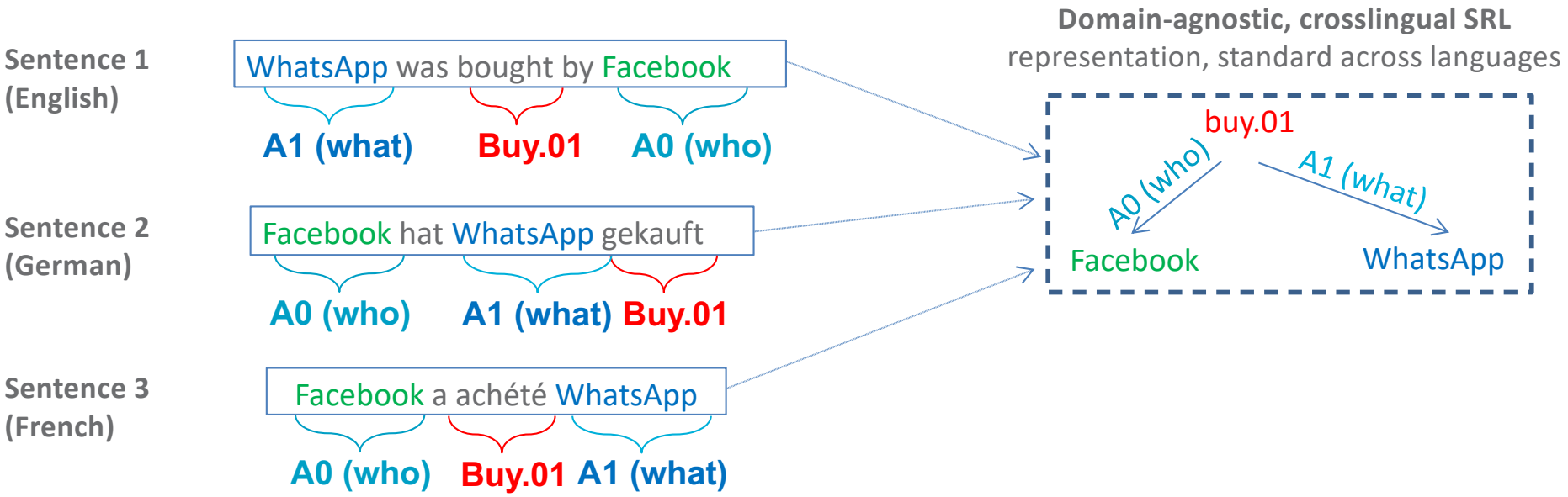
- Different annotation schemes
  - for different languages
  - even for same language!
- High quality labeled data
  - required for most tasks
- 1 task -> 1 model
  - models are built for one task at a time
  - need extensive retraining
  - downstream applications need different models to be pieced together



Universal Semantic Understanding of Natural Languages



Cross-Lingual Shallow Semantic Parsing



Cross-Lingual Shallow Semantic Parsing

Task: Extract who bought what

Sentence 1 (English)

WhatsApp was bought by Facebook

A1 (what) Buy.01 A0 (who)

Sentence 2 (German)

Facebook hat WhatsApp gekauft

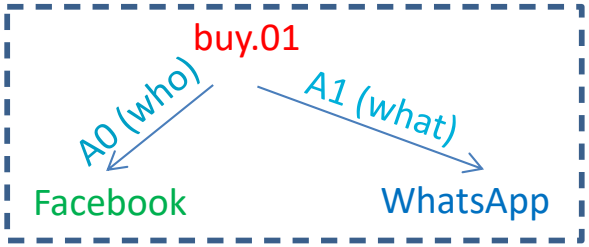
A0 (who) A1 (what) Buy.01

Sentence 3 (French)

Facebook a achété WhatsApp

A0 (who) Buy.01 A1 (what)

Domain-agnostic, crosslingual SRL representation, standard across languages



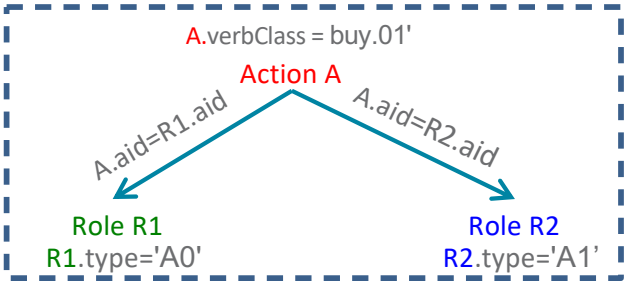
Domain-specific extractor: Select:

- Verb with sense „buy“
- A0 argument as Buyer
- A1 argument as Purchase

Sentence	Verb	Buyer	Purchase
1	buy.01	Facebook	WhatsApp
2	buy.01	Facebook	WhatsApp
3	buy.01	Facebook	WhatsApp

Cross-Lingual Shallow Semantic Parsing in AQL

WhatsApp was bought by Facebook.
Facebook hat WhatsApp gekauft.
Facebook a achété WhatsApp.



Action view

aid	verb	verbClass	…
0	bought	buy.01	…
1	gekauft	buy.01	…
2	achété	buy.01	…

Role view

aid	type	head	description	…
0	A0	Facebook	Buyer	…
0	A1	WhatsApp	Thing bought	…
1	A0	Facebook	Buyer	…
1	A1	WhatsApp	Thing bought	…
2	A0	Facebook	Buyer	…
2	A1	WhatsApp	Thing bought	…

Expanded Shallow Semantic Parser output exposed as AQL views

AQL implementation of cross-lingual extractor

```
create view Acquisition as
select A.verb as buy_verb,
       R1.head as buyer,
       R2.head as purchase
from Action A, Role R1, Role R2
 where
    Equals(A.aid,R1.aid) and
    Equals(A.aid,R2.aid) and
    Equals(R1.type, 'A0') and
    Equals(R2.type, 'A1') and
    Equals(A.verbClass, 'buy.01');
```

Acquisition view

buy_verb	buyer	purchase
bought	Facebook	WhatsApp
gekauft	Facebook	WhatsApp
achété	Facebook	WhatsApp

Result of AQL Extractor

Cross-lingual Information Extraction

Enter text here:

My friend bought the iPhone6 yesterday. I won't buy an iPhone though. I instead am going to save up and get the S8 when it comes out next year.

如果我找到工作的话，我会在今年年底买 一个新iphone7。

Mein Kollege meinte letztens, dass er sich das Nexus5X kaufen wird. I kaufe mir ganz sicher kein iPhone.

Parcontre, Jean-Luc veux acheter le HTC One.

Enter AQL here:

```
create dictionary PurchaseVerbs with case insensitive as
    ('buy.01', 'purchase.01', 'acquire.01', 'get.01');

create view Relation as
select A.polarity as will_buy, A.verb as buy_verb, R2.head
    as purchase
 from Action A, Role R1, Role R2
 where
  Equals(A.aid,R1.aid) and
  Equals(A.aid,R2.aid) and
  Equals(R1.type, 'A0') and
  Equals(R2.type, 'A1') and
  MatchesDict('PurchaseVerbs',A.verbClass);
```

Annotated text:

My friend bought (BUY_VERB) the iPhone6 (PURCHASE) yesterday. I won't buy (BUY_VERB) an iPhone (PURCHASE) though. I instead am going to save up and get (BUY_VERB) the S8 (PURCHASE) when it comes out next year.

如果我找到工作的话，我会在今年年底 买 (BUY_VERB) 一个新 iphone7 (PURCHASE) 。

Mein Kollege meinte letztens, dass er sich das Nexus5X (PURCHASE) kaufen (BUY_VERB) wird. I kaufe (BUY_VERB) mir ganz sicher kein iPhone (PURCHASE) .

Parcontre, Jean-Luc veux acheter (BUY_VERB) le HTC (PURCHASE) One.

Extractions:

BUY_VERB	PURCHASE	WILL_BUY
bought	iPhone6	affirmative
buy	iPhone	negative
get	S8	affirmative
买	iphone7	affirmative
kaufen	Nexus5X	affirmative
kaufe	iPhone	negative
acheter	HTC	affirmative

https://vimeo.com/180382223

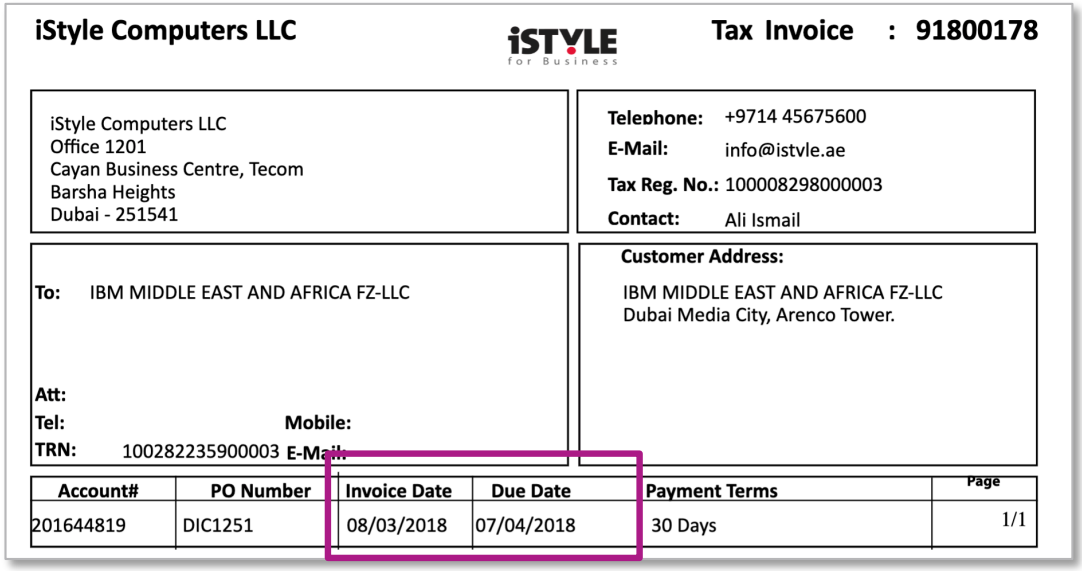
https://github.com/System-T/UniversalPropositions

Table Understanding Example: Invoice Understanding

Task: Extract Invoice Date and Due Date

The classical approach of removing document structure followed by analysis of plain text is not suitable

Requires Document & Table Structures



iStyle Computers LLC iSTYLE for Business **Tax Invoice : 91800178**

iStyle Computers LLC
Office 1201
Cayan Business Centre, Tecom
Barsha Heights
Dubai - 251541

Telephone: +9714 45675600
E-Mail: info@istyle.ae
Tax Reg. No.: 100008298000003
Contact: Ali Ismail

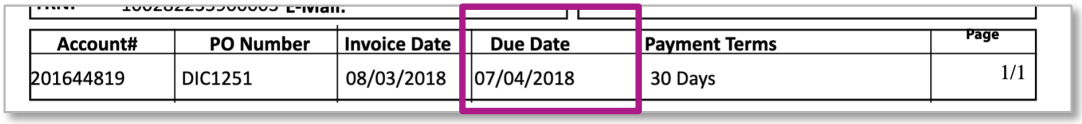
To: IBM MIDDLE EAST AND AFRICA FZ-LLC

Att:
Tel: **Mobile:**
TRN: 100282235900003 **E-Mail:**

Customer Address:
IBM MIDDLE EAST AND AFRICA FZ-LLC
Dubai Media City, Arenco Tower.

Account#	PO Number	Invoice Date	Due Date	Payment Terms	Page
201644819	DIC1251	08/03/2018	07/04/2018	30 Days	1/1

Invoice Due Date Extraction: Implementation in AQL



Account#	PO Number	Invoice Date	Due Date	Payment Terms	Page
201644819	DIC1251	08/03/2018	07/04/2018	30 Days	1/1

Table Cell contains Date

AND Column Header contains Due Date clue → DueDate Entity

AQL Implementation of domain-specific extractor

```
create dictionary DueDate_Dict as
          ('Due Date', 'Due', 'Invoice Due', ...);

create view DueDate as
select
      D.date
from
      Date D,
      TableCell C
where
      ContainsDict('DueDate_Dict', C.columnHeader) and
      Contains(C.cell, D.date);
```

- Named Entity Library
- Table Understanding Library

Domain-agnostic Building Blocks, reusable across use-cases and domains

AQL Text Operators:
Dictionary Matching, Join on Spans

Invoice Understanding: Other Example Tasks Benefiting from Table Understanding

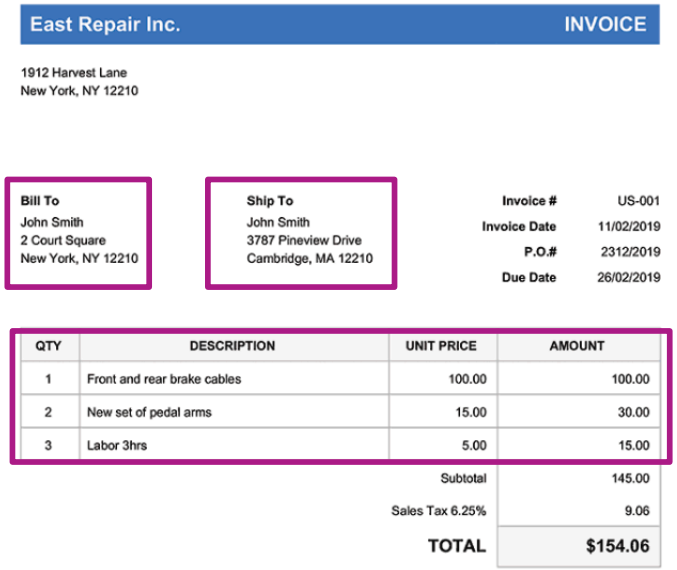
Entity extraction

- Billing Address
- Shipping Address
- Item Quantity, Description, Unit Price & Amount

4-ary Relation Extraction

- Associate Quantity, Description, Unit Price and Amount corresponding to each Part

Same challenges as previous example



East Repair Inc. INVOICE

1912 Harvest Lane
New York, NY 12210

Bill To
John Smith
2 Court Square
New York, NY 12210

Ship To
John Smith
3787 Pineview Drive
Cambridge, MA 12210

Invoice #	US-001
Invoice Date	11/02/2019
P.O.#	2312/2019
Due Date	26/02/2019

QTY	DESCRIPTION	UNIT PRICE	AMOUNT
1	Front and rear brake cables	100.00	100.00
2	New set of pedal arms	15.00	30.00
3	Labor 3hrs	5.00	15.00
		Subtotal	145.00
		Sales Tax 6.25%	9.06
		TOTAL	**$154.06**

/endTutorial

SystemT KDD 2019 Tutorial on GitHub

SystemT Homepage

SystemT MOOC