

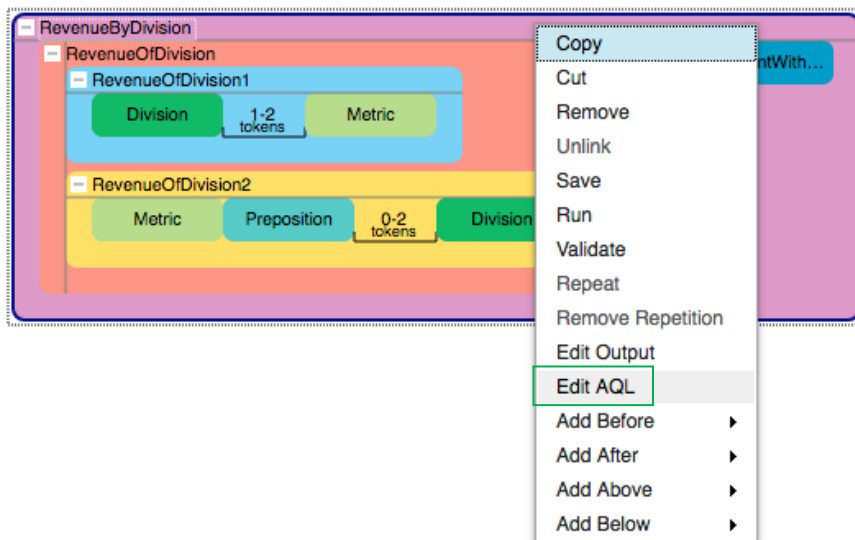
# SystemT Advanced Rule Editor Lab

## Part 2: Creating Advanced Extractors and Using AQL

You will now learn about more advanced functionality, by continuing to work with the UI components as well as working directly with the AQL code.

Before you begin: Explore the AQL code generated by Part 1

1. Right click on the *RevenueByDivision* extractor and select 'Edit AQL' from the dropdown menu:



The canvas view will change to display the AQL file of each extractor. You can return to the canvas view by clicking the yellow back arrow in the top right menu:

```

1 module tauser_Financial__Export;
2
3 /**
4  * @author tauser (via Text Analytics Web Tooling)
5  */
6
7  -----
8  -- Rules for the extractor 'RevenueByDivision' --
9  -----
10
11 -- Extract the text from the document that conforms
12 -- to the following sequence pattern
13 -- Final view representing the extractor 'RevenueByDivision' as per the
14 -- output specifications
15 create view RevenueByDivision as
16   extract pattern (<R.RevenueOfDivision>) <Token>{0,35} (<A.AmountWithUnit>)
17   return group 0 as RevenueByDivision
18   and group 1 as RevenueOfDivision
19   and group 2 as AmountWithUnit
20   from RevenueOfDivision R,
21   AmountWithUnit A;
22
23 -- Export the final view and dictionaries so that they can
24 -- be referenced in other modules
25 export view RevenueByDivision;
26

```

2. Click around and look at the AQL behind each of the extractors you had created in part 1.

As an example, note how the “extract pattern” statement in RevenueOfDivision1.aql matches the pattern you had constructed visually to create this extractor:

```

create view _RevenueOfDivision1_Projection as
extract pattern (<D.Division>) <Token>{1,2} (<M.Metric>)
return group 0 as RevenueOfDivision
and group 1 as Division
and group 2 as Metric
from Division D,
Metric M;

```

As another example, in SentenceBoundary.aql, note how the filtering of matches from the Abbreviations extractor is done using a Minus statement and the built-in Overlaps function:


```





create view SentenceBoundary as
(select S1.SentenceBoundary
from _SentenceBoundary_0 S1)
minus
(select S1.SentenceBoundary
from _SentenceBoundary_0 S1,
Abbreviations A
where Overlaps(S1.SentenceBoundary, A.Abbreviations));

```

3. Note that the settings pane has now changed to be the AQL Resources pane, showing the three dictionaries you had created, as well as their contents:

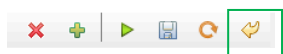
AQL Resources			
Icon	Resource Name	Type	
mdar	Abbreviations.dict	Dictionary	
mdar	Division.dict	Dictionary	
mdar	Metric.dict	Dictionary	
mdar	Preposition.dict	Dictionary	







Newest

- Software
- Hardware
- Global Business Services
- Global Technology Services

When you have finished examining the AQL generated from Part 1, click the yellow back arrow in the top right menu to return to the canvas.

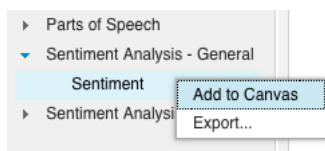


## Lab Scenario 2: Financial Sentiment Analysis

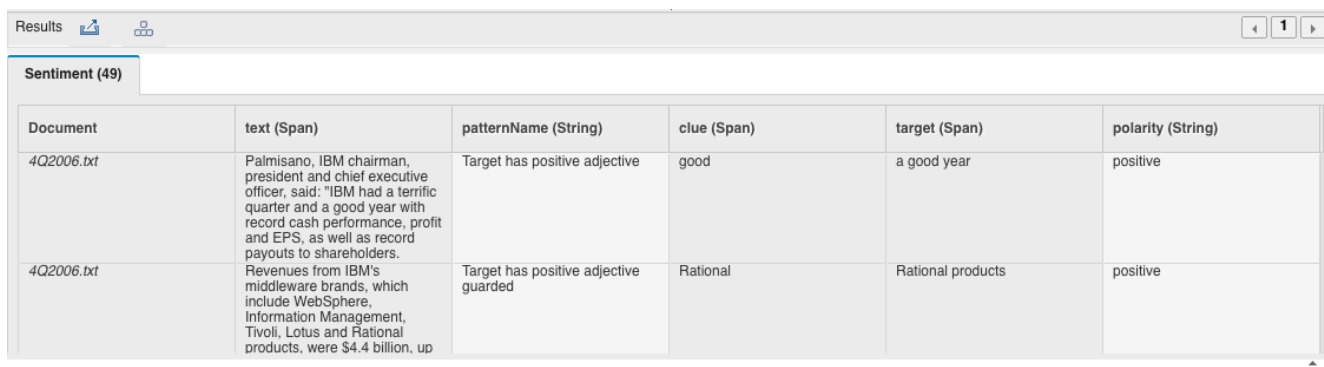
### Step 1: Using a prebuilt extractor

Let's continue to play the role of a data analyst, who is now interested in analyzing the sentiment present in the company earnings report. We will begin by using the prebuilt sentiment extractor.

1. In the projects and extractors pane, click the "Extractors" tab then navigate to the "Sentiment Analysis – General" and expand it.
2. Right click on "Sentiment" and click on Add to Canvas.



1. You will see the extractor show up on the canvas.
2. Click on the extractor and run it, and examine the results:



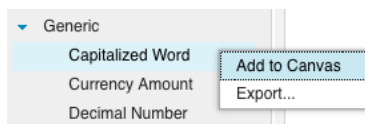
Document	text (Span)	patternName (String)	clue (Span)	target (Span)	polarity (String)
4Q2006.txt	Palmisano, IBM chairman, president and chief executive officer, said: "IBM had a terrific quarter and a good year with record cash performance, profit and EPS, as well as record payouts to shareholders.	Target has positive adjective	good	a good year	positive
4Q2006.txt	Revenues from IBM's middleware brands, which include WebSphere, Information Management, Tivoli, Lotus and Rational products, were \$4.4 billion, up	Target has positive adjective guarded	Rational	Rational products	positive

If you look through the extractor output you will notice a number of annotations that you do not actually wish to pick up – false positives. Let's begin by addressing annotations that look like 'Rational products', where the sentiment clue ("rational") is actually part of a product name.

### Step 2: Filtering false positives using a pattern

We might address annotations like 'Rational products' in multiple ways – in this case, we are going to create an extractor to filter out such false positive cases.

1. First add a prebuilt extractor, **Capitalized Word**, to the canvas. You will find it in the Generic category.



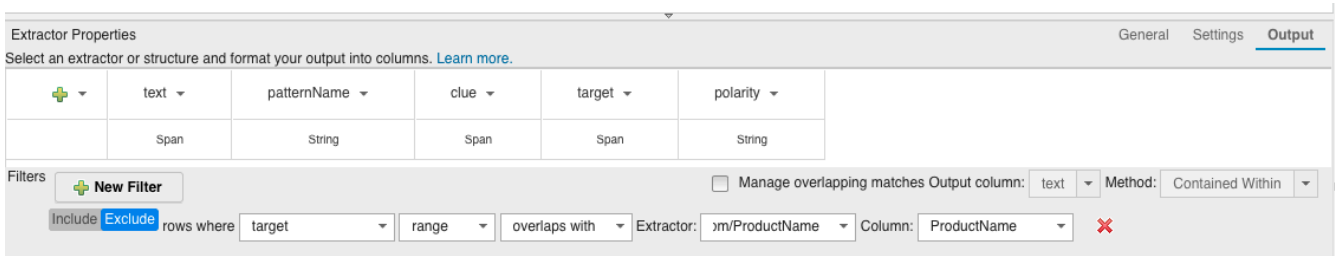
2. Then make a dictionary extractor called **Products** and add to it the terms "product", "products", and "software".
3. Combine Products and Capitalized Word by dragging them next to each other to make a new extractor,

and call it **ProductName**.



We will now use **ProductName** to filter the output of the Sentiment extractor.

1. Click on the Sentiment Extractor you created in step 1.
2. In the Extractor Properties pane on the lower part of the screen, select the Output tab. Towards the bottom of the pane click on New Filter. Set this filter to Exclude rows where target range overlaps with Extractor: ProductName Column: ProductName.



3. Run the **Sentiment** extractor again and note that the output no longer picks up annotations like ‘Rational products.’
4. Save the **Product** dictionary and the **ProductName** extractor to the **Financial** category, to be added to your previously created extractors. Also save your **Sentiment** extractor.

### Step 3: Modifying pre-built extractor using available customization points

If you continue to examine the results, you will come across an annotation like this:

Sentiment (21)					
Document	text (Span)	patternName (String)	clue (Span)	target (Span)	polarity (String)
4Q2006.txt	performance, profit and EPS, as well as record payouts to shareholders. Intellectual property and custom development income increased to \$241 million compared with \$228 million a year ago.	Target has positive adjective	Intellectual	Intellectual property and custom development income increased to \$241 million	positive

The **Sentiment** extractor is picking up on the clue of “intellectual” as an adjective with positive polarity. There are several ways to address this false positive and we are going to demonstrate one particular approach that demonstrates another capability of the editor: modifying the pre-built Sentiment extractor using available customization points.

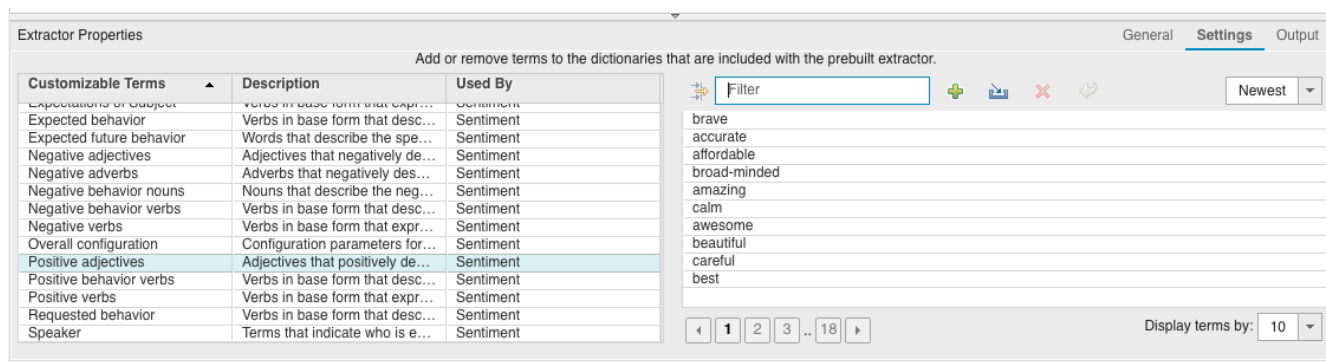
We will start by editing the dictionary used in **Sentiment** extractor.

1. Select the **Sentiment** extractor and click on the Settings tab in the extractor properties pane. Notice that this pre-built extractor has a set of customizable dictionaries, such as “Speaker” and “Negative adjectives.”

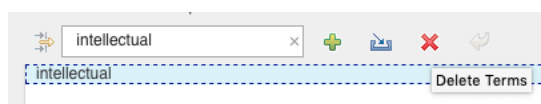
We will modify one of these dictionaries so that the extractor’s performance is in better line with our expectations. The Sentiment extractor explains the reason for picking up on a particular span as a clue in the patternName column. Here, the reason is, “Target has positive adjective,” meaning that the clue, “intellectual,” is being identified as an adjective with positive polarity. However, in the text it is actually a component of the phrase “intellectual

property,” and should not be identified with positive polarity.

2. In the Settings tab, select the “Positive adjectives” dictionary to see its set of customizable terms.



3. Use the Filter box to search for “intellectual” and delete that term:

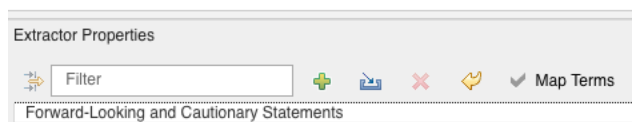


4. Run the **Sentiment** extractor again and review the results, noting that the previously identified false positive is no longer being picked up.
5. Save the modified **Sentiment** extractor to the **Financial** category, overwriting your previous version.

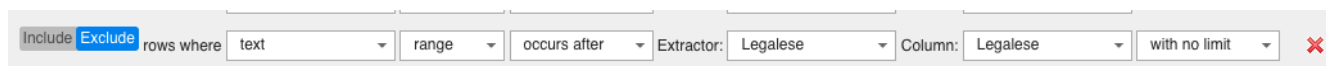
#### Step 4: Limiting document scope

Another type of false positive annotation comes from the boilerplate legalese sections that are often present in such documents – in this case, as the section, “Forward-Looking and Cautionary Statements.” The text there is not relevant to our sentiment extractor task and will invariably yield unwanted output. We will limit the scope of our extractor to forbid it from examining the text in that section.

1. Make a dictionary extractor, called **Legalese**, and add to it the term “Forward-Looking and Cautionary Statements”



2. Now add a second filter to the **Sentiment** extractor by clicking the Output pane of the Extractor Properties. Set this filter to Exclude rows where text range occurs after Extractor: Legalese Column: Legalese with no limit:

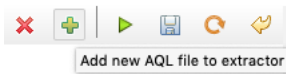


3. Run the **Sentiment** extractor again and examine the results.
4. Save the **Legalese** extractor and the modified **Sentiment** extractor to the **Financial** category.

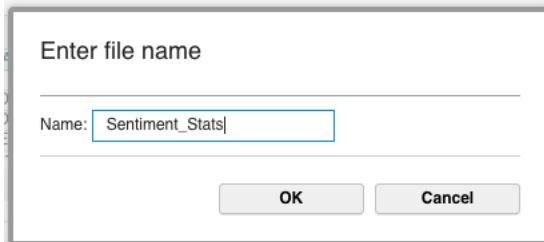
## Step 5: Writing AQL to summarize extractor results

We may wish to gain an overview of the sentiments identified in each document. One way to do this is to count the number of instances of positive and negative sentiment, broken down by the various clue spans that indicated sentiment. To do this, we will shift to writing AQL directly.

1. Open the AQL view by right clicking on the **Sentiment** extractor and selecting Edit AQL.
2. We will begin by adding a new AQL file.



3. Name this new extractor **Sentiment\_Stats**

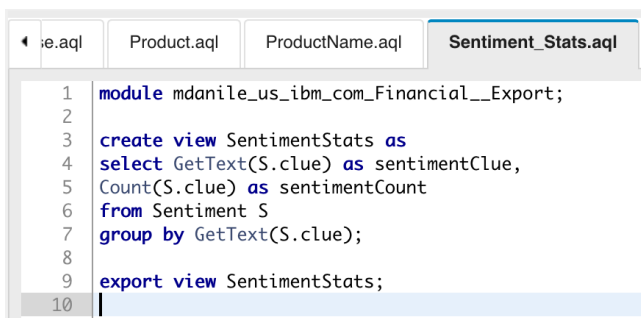


4. A new tab will open, named Sentiment\_Stats.aql. You will see a single line that reads:  
module <module\_name>;
5. Go to the Sentiment.aql tab and copy the first line that starts with the word “module”.
6. Paste that full line (including the semicolon ; ) as the first line in Sentiment\_Stats.aql – *Replacing* the line that was originally there.
7. Copy the below AQL code and paste it on the second line of Sentiment\_Stats.aql. You can also find this code in the file Lab3.aql.



```
create view SentimentStats as
select GetText(S.clue) as sentimentClue,
Count(S.clue) as sentimentCount
from Sentiment S
group by GetText(S.clue);

export view SentimentStats;
```

Your Sentiment\_Stats.aql file should look like this (with your unique module name at the top):



1. Run Sentiment\_Stats.aql by clicking on the green play icon, as before, and examine the results:

Results  			◀ 1 ▶		
CapsWord (1239)	Legalese (5)	Product (87)	ProductName (25)	Sentiment (16)	<b>SentimentStats (10)</b>
Document	sentimentClue (text)	sentimentCount (text)			
4Q2006.txt	good	1			
4Q2007.txt	solid	1			
4Q2008.txt	extremely difficult	1			
4Q2008.txt	solid	1			
4Q2009.txt	delivered	1			
4Q2009.txt	greater	1			
4Q2009.txt	improved	3			
4Q2009.txt	solid	1			
4Q2010.txt	helped	5			
4Q2010.txt	improved	1			

You can see that the view we created is counting the number of mentions of each sentimentClue.

Notice that the results are, grouped per document. Thus you see 2 mentions of “competitive” in 4Q2009.txt and one mention in 4Q2010.txt. Recall that SystemT operates on a document-at-a-time model.

Notice also that we used the built-in function GetText in order to be able to group the text values of the spans together. If we were to omit using that function, the output would merely be a list of spans, with the count necessarily being 1 for each span. Try removing the GetText function to satisfy yourself of the altered output.

### Step 6: AQL for lists and counts

Let’s go one step further. We would now like a summary of which sentiment patterns were present in each document, as well as the list of clues for each pattern.

2. Copy the following code into Sentiment\_Stats.aql *below* your code from the previous step (You can also find this code in the file Lab3.aql):

```
create view SentimentPatterns as
select GetText(S.patternName) as patternName,
       List(GetText(S.clue)) as clues,
       Count(GetText(S.clue)) as countClues
from Sentiment S
group by GetText(S.patternName);
```

```
export view SentimentPatterns;
```

(don’t forget the semicolons at the end!)



3. Run Sentiment\_Stats.aql again:

239	Legalese (5)	Product (87)	ProductName (25)	Sentiment (16)	SentimentPatterns (9)
Document	patternName (text)	clues (text)	countClues (text)		
4Q2006.txt	Target has positive adjective	['good']	1		
4Q2007.txt	Target has positive adjective	['solid']	1		
4Q2008.txt	Target has negative adjective	['extremely difficult']	1		
4Q2008.txt	Target has positive adjective	['solid']	1		
4Q2009.txt	Target does positive	['delivered']	1		
4Q2009.txt	Target has positive adjective	['solid', 'improved']	2		
4Q2009.txt	Target has positive adjective guarded	['greater', 'improved', 'improved']	3		
4Q2010.txt	Target does positive	['helped', 'helped', 'helped', 'helped', 'helped']	5		
4Q2010.txt	Target has positive adjective guarded	['improved']	1		

This demonstrates the use of AQL Lists, which can combine any AQL output of the same type. We see, for example, that 4Q2009.txt had 2 instances of positive sentiment due to the pattern “Target has positive adjective,” and that in particular, the positive adjectives identified were “solid” and “improved.”

**Congratulations!**  
**You have completed the Advanced Rule Editor Lab!**

**Keep going!**

Try out other information extraction tasks, either by working on the canvas or writing AQL!

AQL Documentation may be found here: <https://ibm.biz/BdX3aR> (select the most recent version of BigInsights).