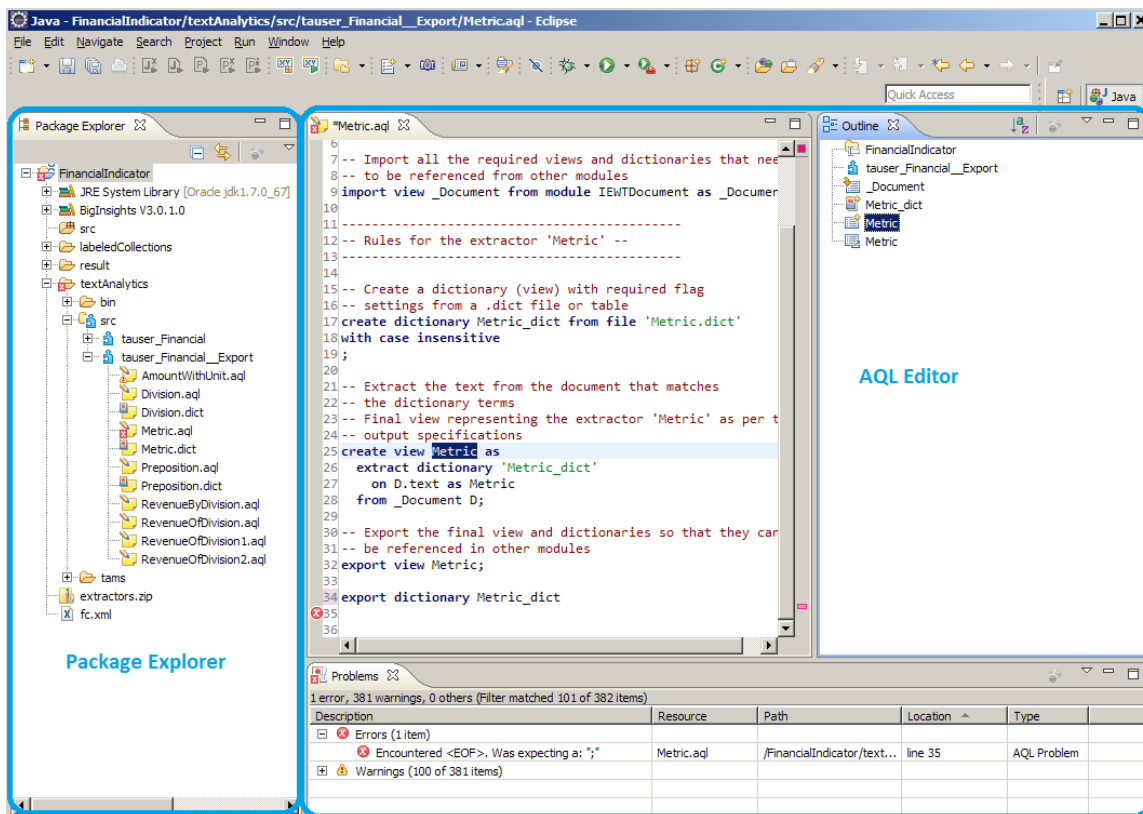


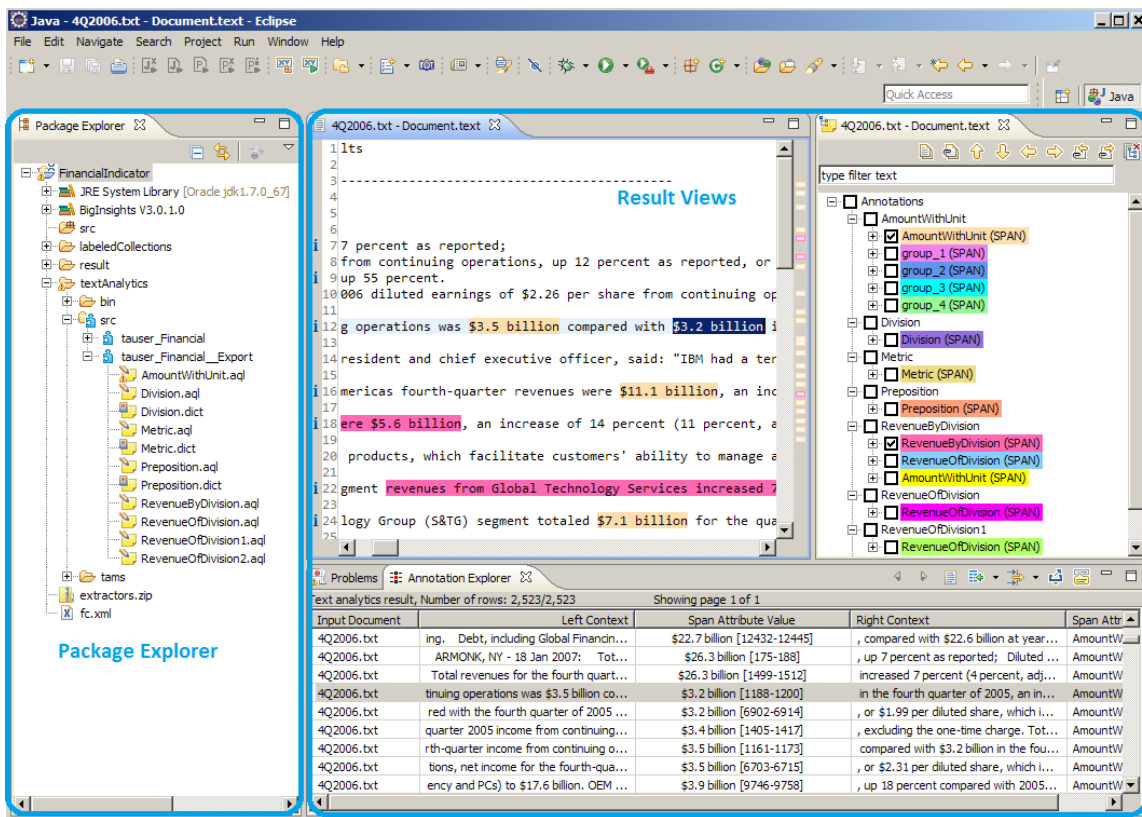
## SystemT Eclipse Tooling Lab

### Before you begin:

In this lab you will explore and learn about the core functionality of the SystemT Eclipse Tooling development environment.

Below are the core UI components of the SystemT Eclipse development environment.





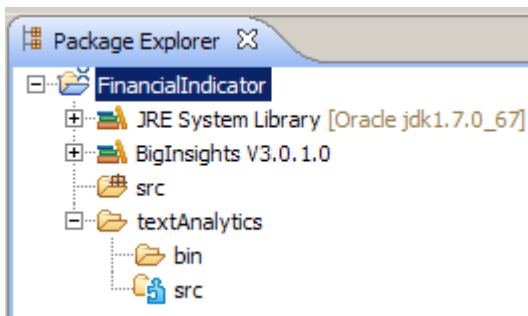
### What you will learn:

- Get started developing AQL rules using the Eclipse Tooling;
- Use advanced tooling functionality to aid in the development process:
  - Explain the **provenance** of results – helps you identify imprecise rules (and improve the accuracy of your extractor)
  - **Annotation Difference** – compare results of different iterations in the development process
  - **Labeled Collections** – prepare an evaluation dataset and use it to evaluate the accuracy and coverage of your extractor
  - **Regular Expression Generator** – automatic discovery of a regular expression based on examples
  - **Pattern Discovery** – helps you identify additional patterns in the text to improve the coverage of your extractor

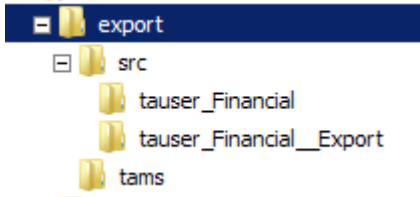
### Procedure

#### Step 1: Import the extractor created using the Web Tooling.

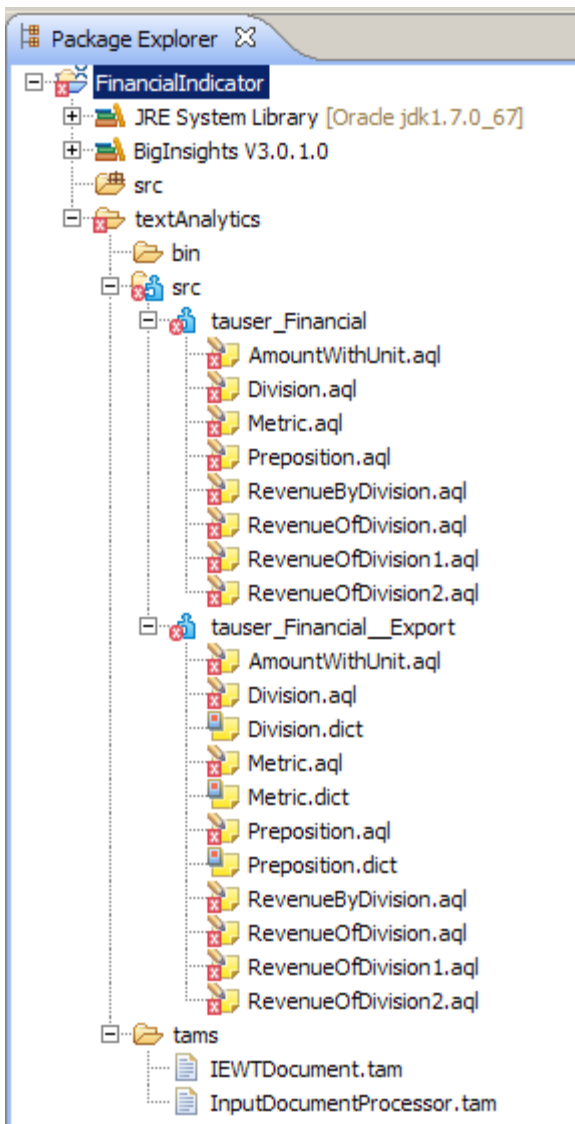
1. In Eclipse, select **File > New > Other > BigInsights > BigInsights Text Analytics Project**.
2. Click **Next**. Fill in the project name as FinancialIndicator and click **Finish**. The project will appear in the Package Explorer:



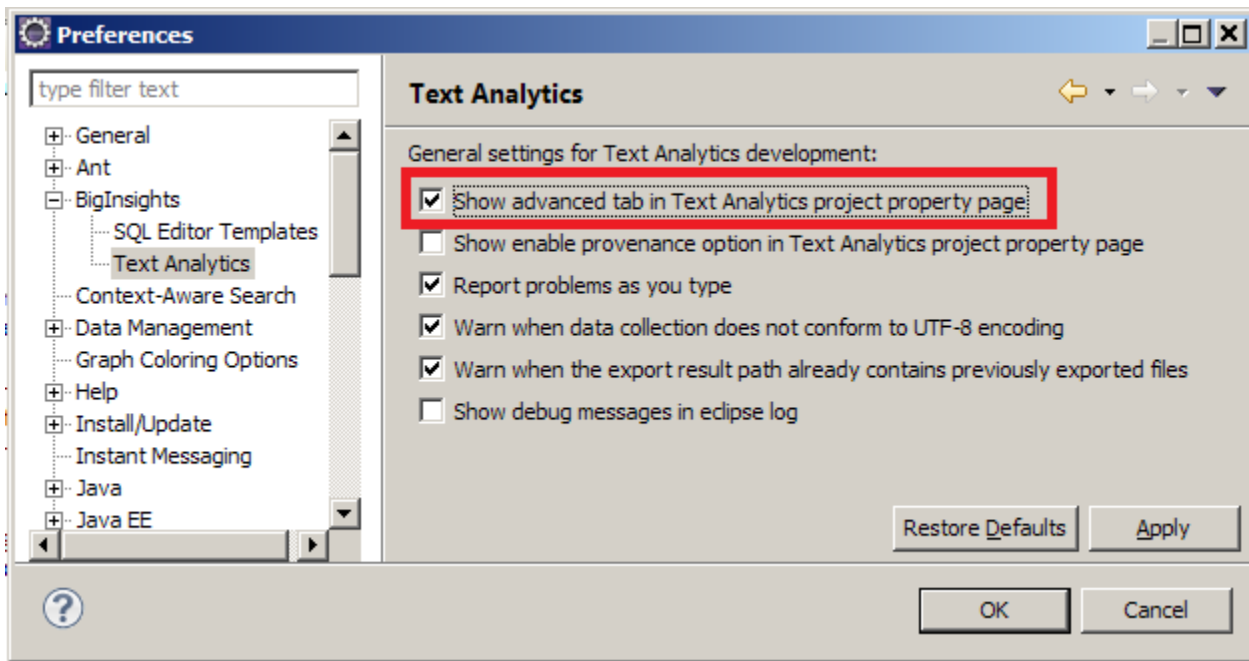
3. On your local file system, navigate to the **exports** directory. Inside the directory, you should see the following folders:



4. Copy the **src** and **tams** folder inside your new FinancialIndicator project (i.e., select the two folders on your local file system, right click to copy them, then back in Eclipse Package Explorer view , right click on the **textAnalytics** folder and select **Paste**). You should see the content of the two folders as follows in the Package Explorer:

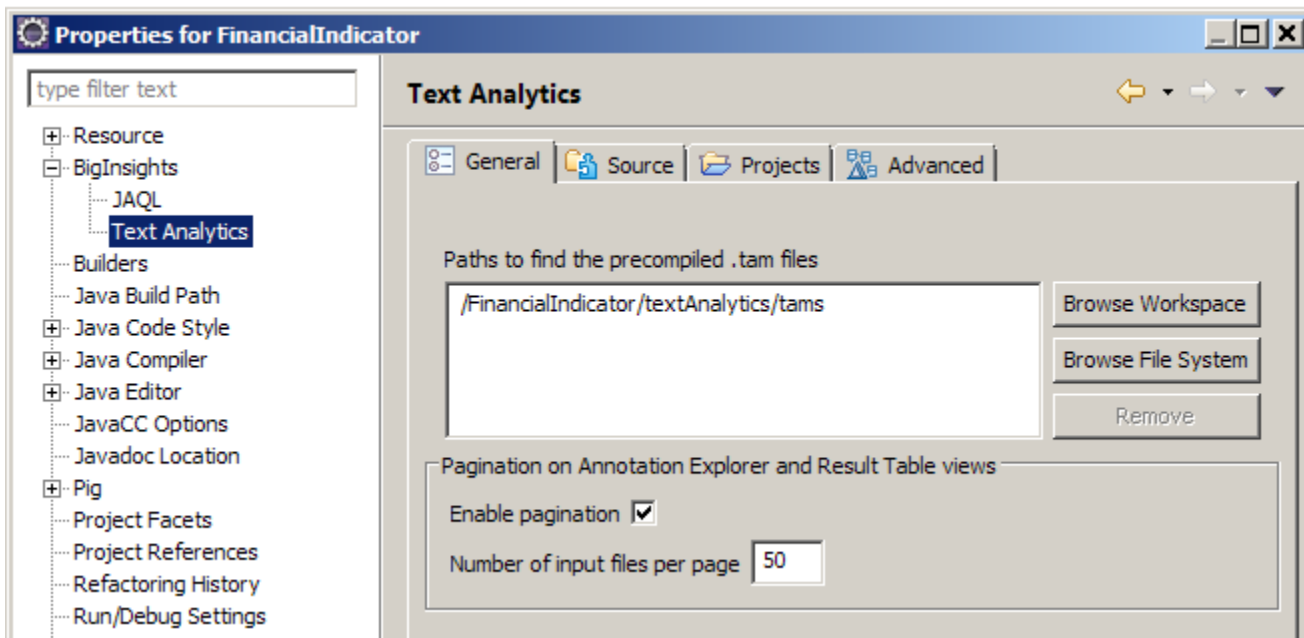


5. First, enable the **Advanced** tab in the Text Analytics properties of your project. In Eclipse main menu, go to **Window > Preferences > BigInsights > Text Analytics** and check **Show advanced tab in Text Analytics Properties** page:



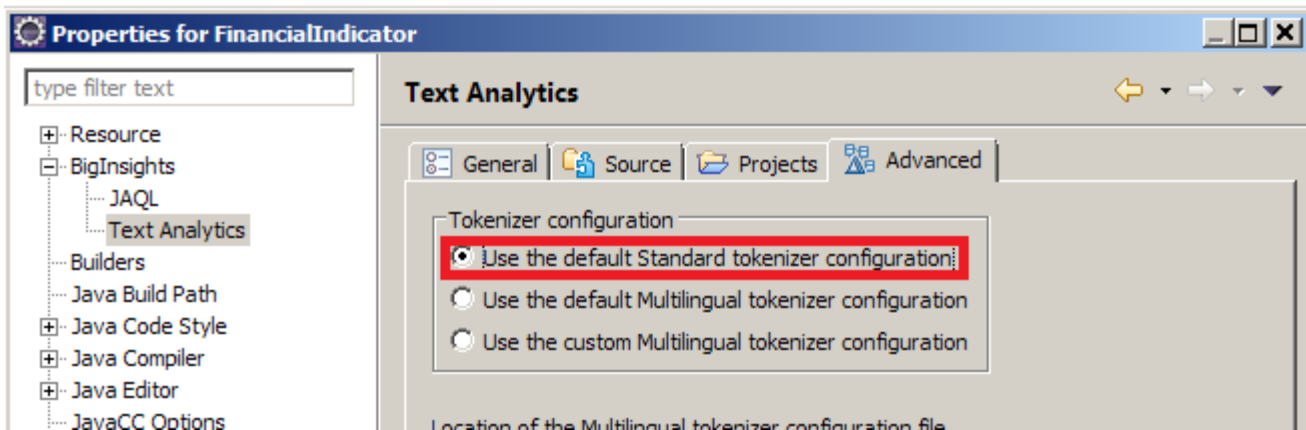
6. Modify the Text Analytics properties of your project to point to the AQL compiled modules in the **tams** folder that you just copied, and to use the Standard tokenizer.

In Package Explorer, right-click on the FinancialIndicator project and select Properties. In the Properties dialog, navigate to **BigInsights > Text Analytics > General**. Under **Paths to find precompiled modules**, select **Brows Workspace** and navigate to the **tams** folder inside your project, selecting the **tams** folder.

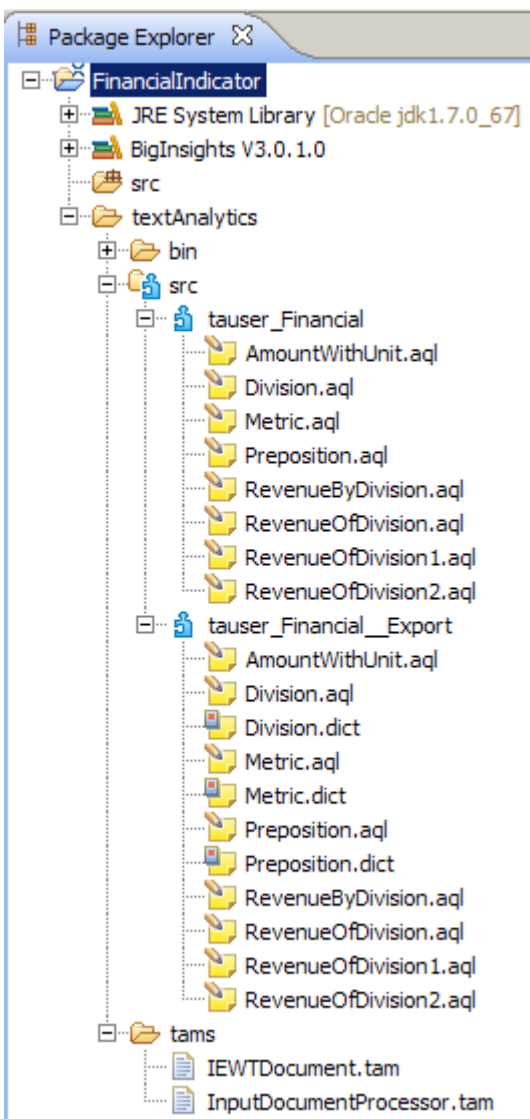


7. Modify the Text Analytics properties of your project to use the Standard tokenizer.

Still in the Text Analytics Properties dialog, go to the **Advanced** tab, and select **Use the default Standard tokenizer configuration**. Click **OK**.



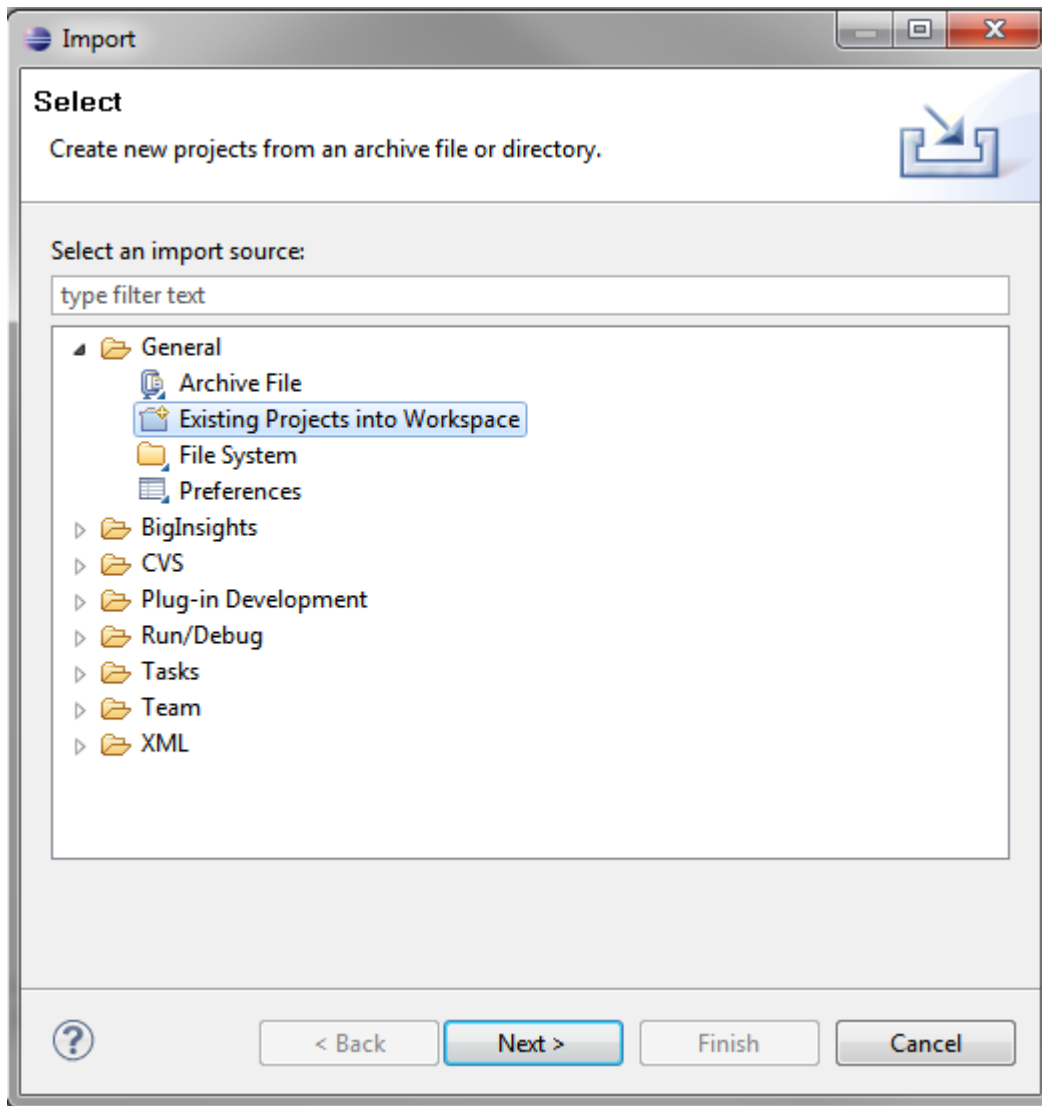
The project is automatically compiled and there should be no error markers on the **src** folder and the .aql files.



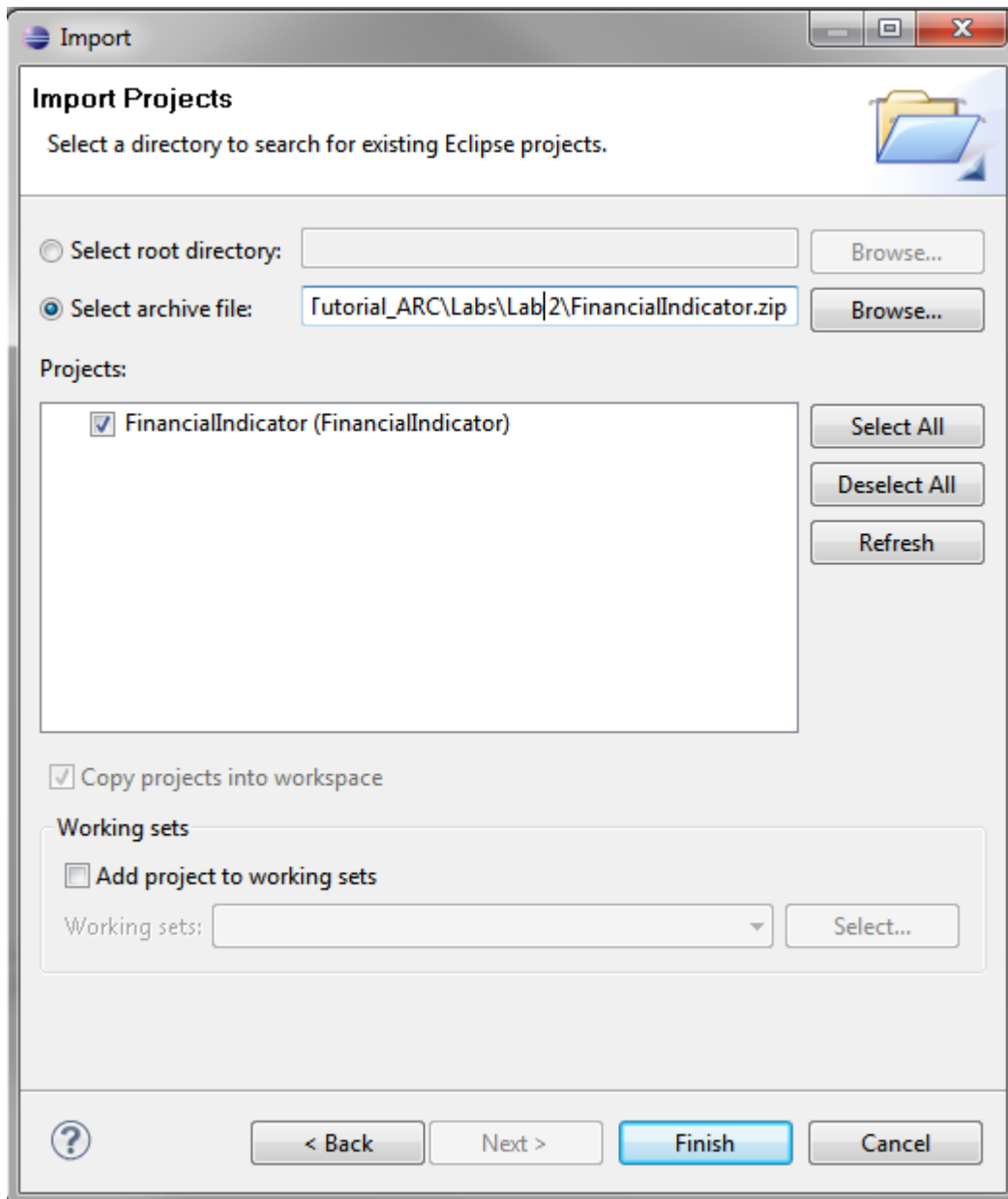
**Step 1 Backup – Perform this step ONLY if you encountered a problem in Step 1. Otherwise (no problem in Step 1), jump to Step 2.**

If you encounter any problems in Step 1, as an alternative procedure, import the basic project supplied with the lab material. The project is in an archive called **FinancialIndicator.zip**.

1. Import the project from the archive into Eclipse. Go to **File > Import > General > Existing Projects into Workspace**:



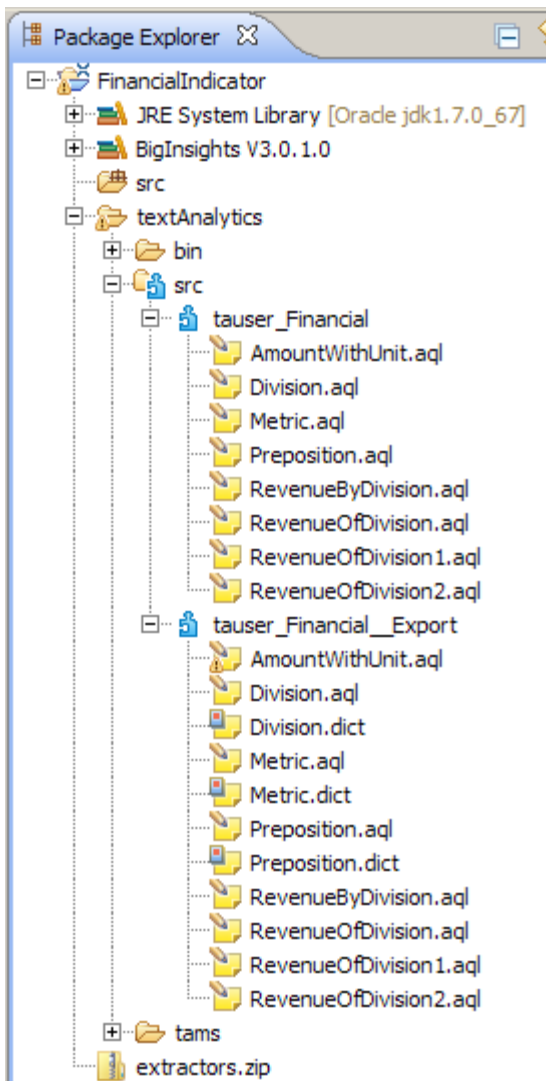
2. Click **Next** and **Browse** to the FinancialIndicator.zip file.



3. Click **Finish**.

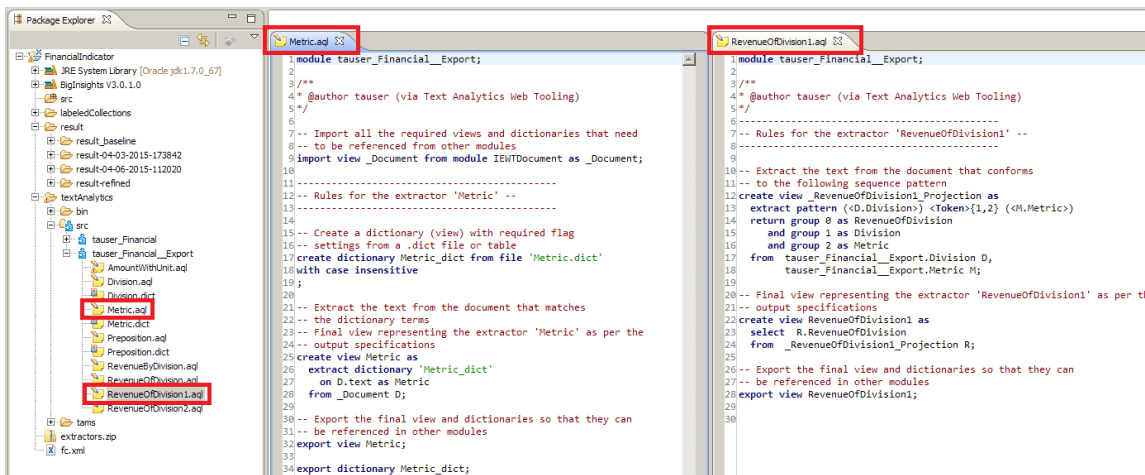
4. Take a moment to verify that the project was imported correctly. In the **Package Explorer**, you should see the same structure as shown in Step 1 (expand various folders as necessary):





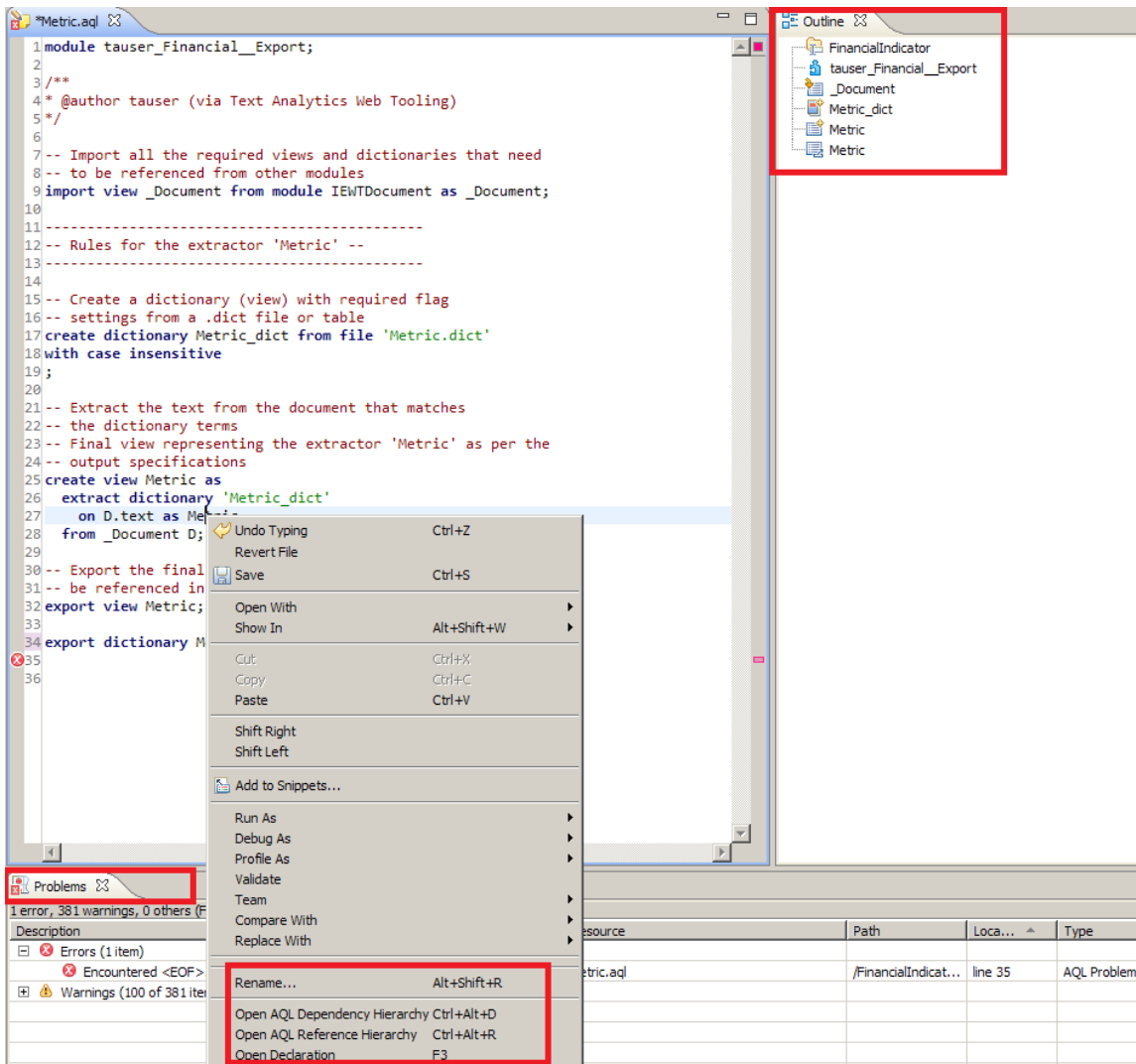
## Step 2: Explore the AQL code.

1. The module **tauser\_Financial\_\_Export** contains the main AQL code generated. There is one AQL file generated for each of the concepts created in the Web Tool. The module **tauser\_Financial** contains output view statements for each of the generated views.
2. In the Package Explorer, double-click on each AQL files in the module **tauser\_Financial\_\_Export** to open it in the AQL Editor. The next screenshot shows two of the AQL files, **Metric.aql** and **RevenueOfDivision1.aql** open in the AQL Editor. These files, along with the two main views defined in these files (metric and respectively RevenueOfDivision1) correspond to the two concepts **Metric** and **RevenueOfDivision1** that you created using the Web Tool.



### 3. The **AQL Editor** offers several useful functionalities for editing AQL:

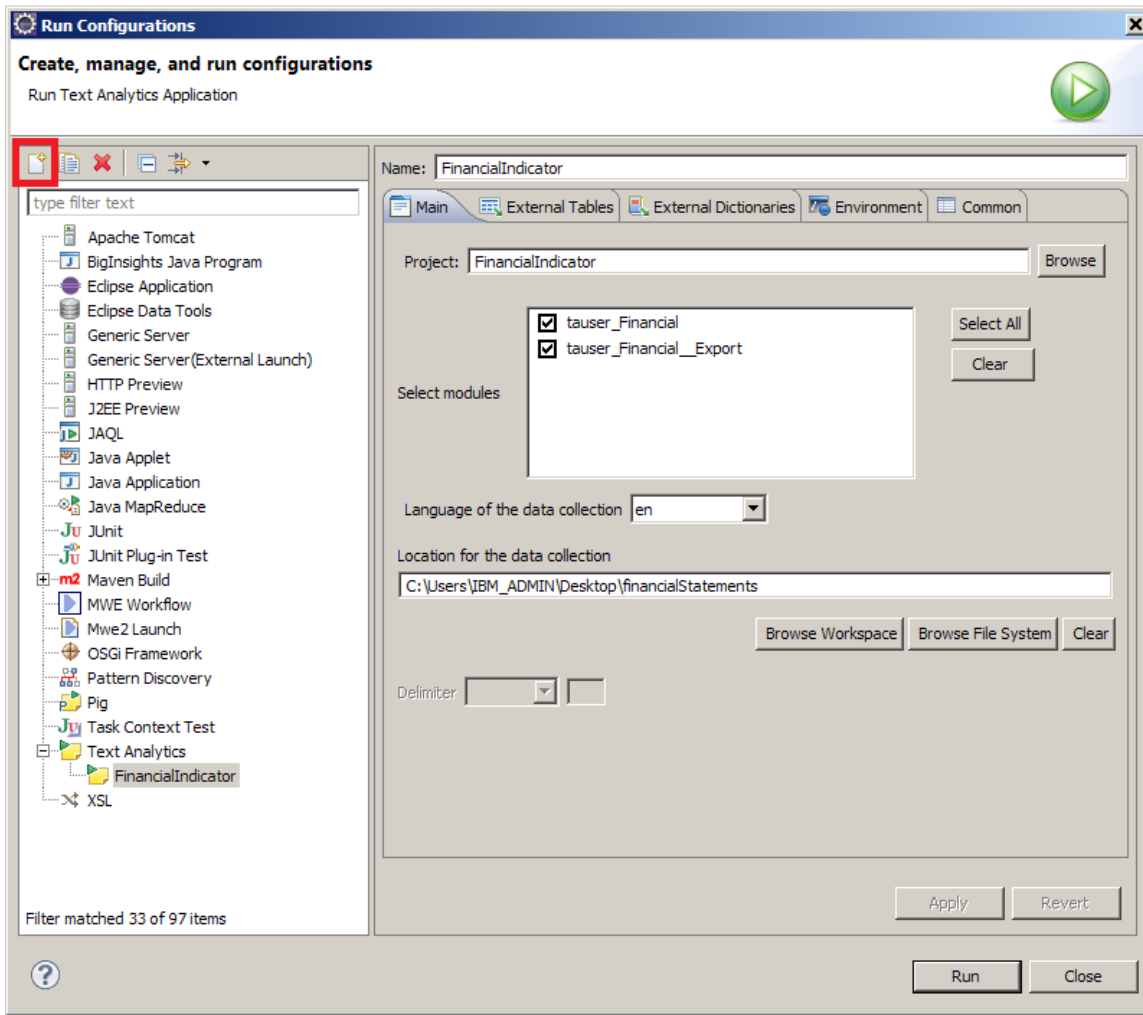
- Syntax highlighting
- Display AQL compiler warnings and errors in the Eclipse **Problems** view
- An Outline view showing all AQL artifacts (views, dictionaries, tables and functions) defined in an AQL file (see the right of the screenshot)
- Refactoring for AQL artifacts (see the Rename context menu item)
- Hyperlink navigation (the Open Declaration context menu item)
- The reference and Dependency hierarchy for AQL views (i.e., which views depend on the current view, and which views the current view depends on)
- Content Assist including auto-completion of view names (using CTRL+Space after selected AQL keywords such as **import**, **export**, **from**).



For more details about the functionality of the AQL Editor, refer to the Eclipse Help, by clicking on **Help > Help Contents > BigInsights > Analyzing text in big data > BigInsights Text Analytics Tooling reference > Extractors > AQL Editor**.

### Step 3: Run the extractor

1. To execute the extractor, create a **Text Analytics Run Configuration**. Go to **Run > Run Configurations**, click on **Text Analytics** and create a new configuration by clicking on the top right button:



2. Fill in the run configuration as shown above, navigating to the set of financial statements to set up the data collection. Click **Run**.

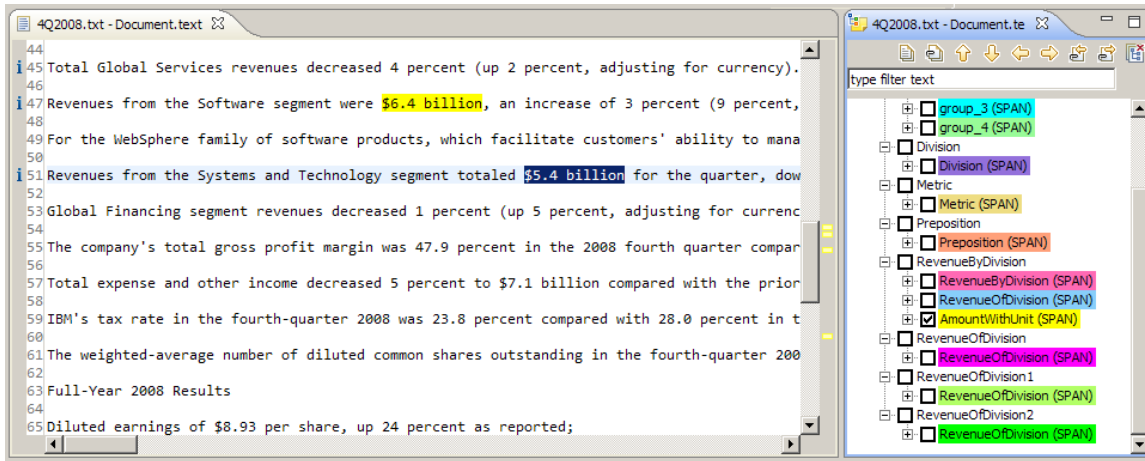
3. All spans extracted appear in the **Annotation Explorer**:

Annotation Explorer

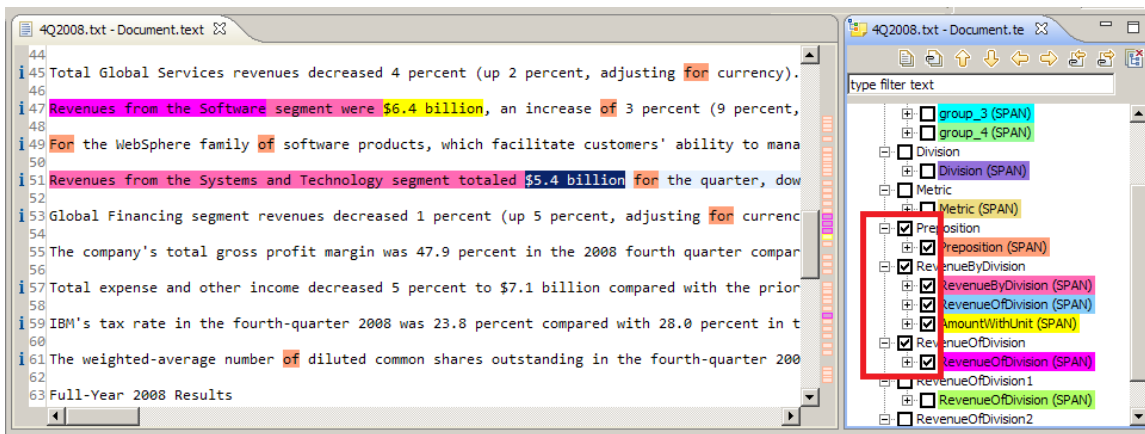
Text analytics result, Number of rows: 2,605/2,605 Showing page 1 of 1

Input Document	Left Context	Span Attribute Value	Right Context	Span Attribute Name
4Q2008.txt	eased 5 percent (flat, adjusting for ...	\$4.7 billion [3601-3613]	. IBM signed services contracts total...	RevenueByDivision.AmountWithUnit
4Q2008.txt	eased 5 percent (flat, adjusting for ...	\$4.7 billion [3601-3613]	. IBM signed services contracts total...	RevenueByDivision.AmountWithUnit
4Q2008.txt	erspective, the America's full-year r...	\$42.8 billion [8978-8991]	, an increase of 4 percent as report...	AmountWithUnit.AmountWithUnit
4Q2008.txt	gement, Tivoli, Lotus and Rational pr...	\$5.2 billion [4503-4515]	, up 4 percent versus the fourth qua...	AmountWithUnit.AmountWithUnit
4Q2008.txt	s from the Systems and Technology ...	\$5.4 billion [5583-5595]	for the quarter, down 20 percent (1...	AmountWithUnit.AmountWithUnit
4Q2008.txt	s from the Systems and Technology ...	\$5.4 billion [5583-5595]	for the quarter, down 20 percent (1...	RevenueByDivision.AmountWithUnit
4Q2008.txt	1 percent (1 percent, adjusting for ...	\$5.5 billion [3024-3036]	. OEM revenues were \$615 million, d...	AmountWithUnit.AmountWithUnit
4Q2008.txt	ar over year. SG&A expense decrea...	\$5.8 billion [6970-6982]	. RD&E expense of \$1.5 billion decre...	AmountWithUnit.AmountWithUnit
4Q2008.txt	sting for currency). Total services si...	\$57.2 billion [9849-9862]	. Software segment revenues in 200...	AmountWithUnit.AmountWithUnit
4Q2008.txt	sting for currency). Total services si...	\$57.2 billion [9849-9862]	. Software segment revenues in 200...	RevenueByDivision.AmountWithUnit
4Q2008.txt	rency. Revenues from the Softwar...	\$6.4 billion [4207-4219]	, an increase of 3 percent (9 percen...	AmountWithUnit.AmountWithUnit
4Q2008.txt	rency. Revenues from the Softwar...	\$6.4 billion [4207-4219]	, an increase of 3 percent (9 percen...	RevenueByDivision.AmountWithUnit
4Q2008.txt	crease of 7 percent at actual rates (...	\$6.6 billion [3908-3920]	, adjusting for currency). Long-term ...	AmountWithUnit.AmountWithUnit

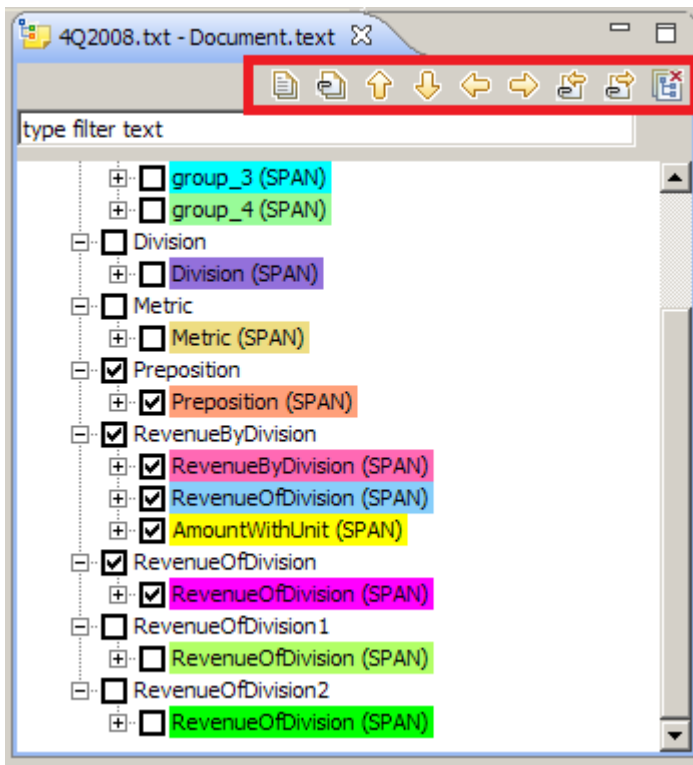
4. In the Annotation Explorer, double click any span. The **Result Editor** appears, along with the **Result Tree** view. The span you clicked on is highlighted in the original document text. In this screenshot, the **Result Editor** is to the left and the **Result Tree** is to the right:



5. In the **Result Tree**, check a few other view/attribute names to see the spans highlighted in the **Result Editor**:



6. On your own, click the different buttons at the top of the **Result Tree** to understand the other available functionality:



7. On your own, explore the rest of the buttons at the top of the **Annotation Explorer** view.

**Hint:** About the **\_Document** view.

Observe that all basic extraction (including extract regular expression and extract dictionary statements, and also the extract pattern statement that contains literals) are done on the view **\_Document** imported from module IEWTDocument, as can be seen for example in `tauser_Financial__Export/Metric.aql`.

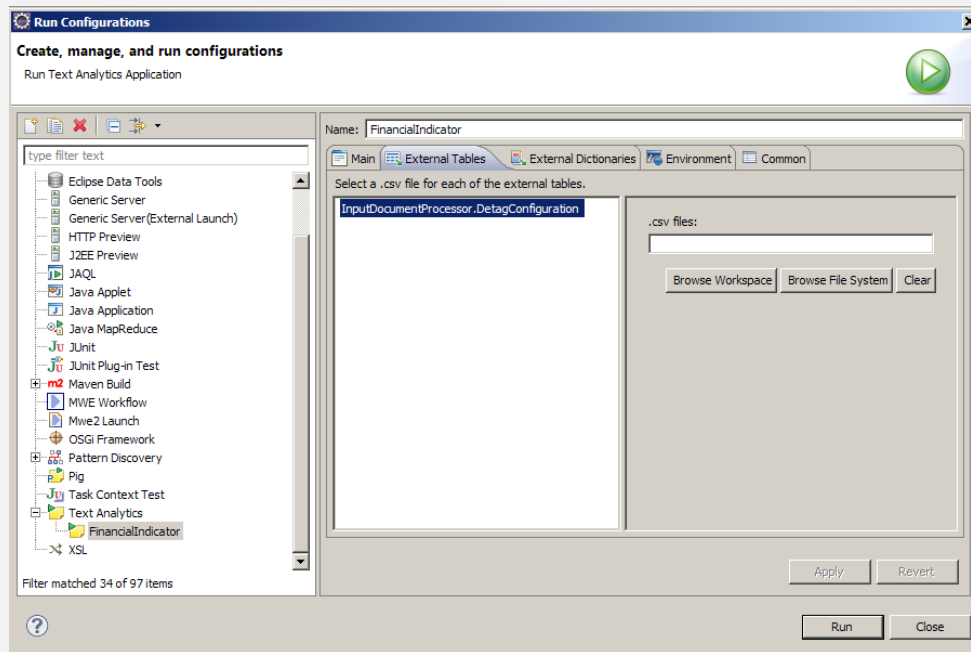
```

Metric.aql
1 module tauser_Financial__Export;
2
3 /**
4  * @author tauser (via Text Analytics Web Tooling)
5  */
6
7 -- Import all the required views and dictionaries that need
8 -- to be referenced from other modules
9 import view _Document from module IEWTDocument as _Document;
10
11 -----
12 -- Rules for the extractor 'Metric' --
13 -----
14
15 -- Create a dictionary (view) with required flag
16 -- settings from a .dict file or table
17 create dictionary Metric_dict from file 'Metric.dict'
18 with case insensitive
19 ;
20
21 -- Extract the text from the document that matches
22 -- the dictionary terms
23 -- Final view representing the extractor 'Metric' as per the
24 -- output specifications
25 create view Metric as
26   extract dictionary 'Metric_dict'
27   on D.text as Metric
28   from _Document D;
29

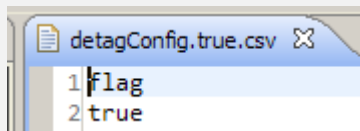
```

This is because the Web Tooling simplifies the specification of the input text by allowing the user to specify

whether the input text contains HTML tags (and hence the content should be first detagged). You can control the detagging specification using the external table `InputDocumentProcessor.DetagConfiguration`, which can be configured in the Text Analytics Run Configuration by pointing to an appropriate CSV file.



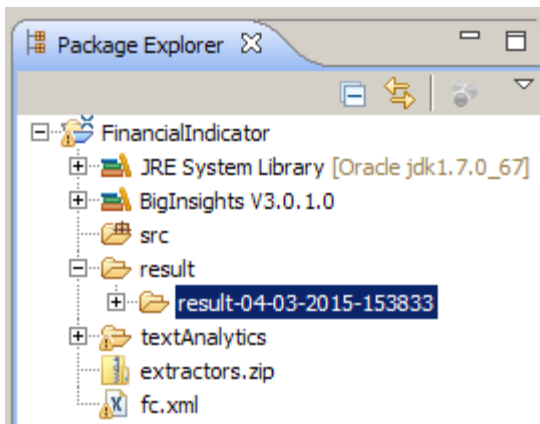
By default, input text detagging is turned off (the `InputDocumentProcessor.DetagConfiguration` external table is not set to any .csv file). If you would like to turn on detagging, point `InputDocumentProcessor.DetagConfiguration` to a CSV file with the following two rows:



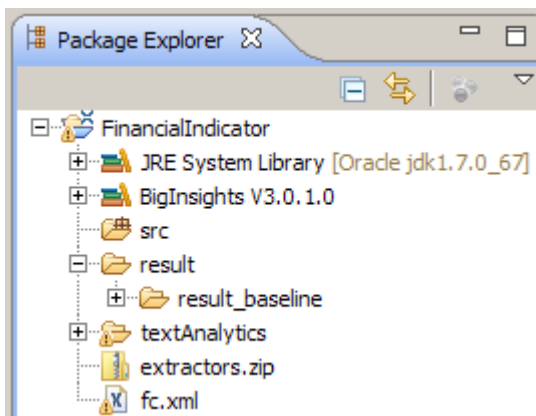
#### Step 4: Save the results

Before we look at the results in more detail, we will tag this result set with the label “result\_baseline”, because later on, once we refine our extractor, we can use the **Annotation Difference Tool** to evaluate how the quality of results has improved over time.

1. In the **Package Explorer**, navigate to the results folder, and locate the result folder with the most recent timestamp (it should show up last in the list). The Eclipse Tools store all results obtained during extractor development in this folder, tagged as “result-`<timestamp>`”, where `<timestamp>` is of the form MM-DD-YYYY-HHMMSS.

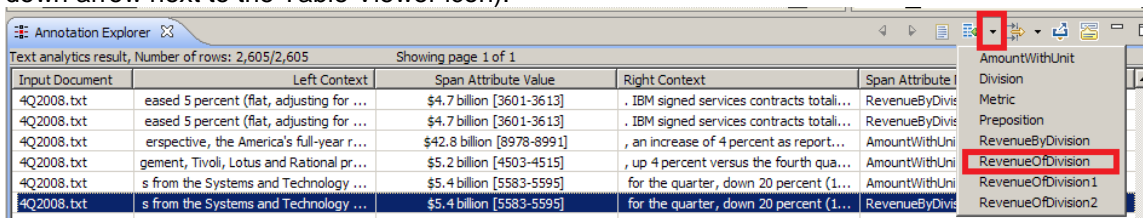


2. Locate the most recent folder, right click, and in the context menu, select **Refactor > Rename**. Then change the name to “result\_baseline”. The folder should now appear like this:



### Step 5: Using the Provenance tool

1. Return to the **Annotation Explorer**, and open the **RevenueOfDivision Table View** as shown (click on the down arrow next to the Table Viewer icon):



The **RevenueOfDivision** opens in the **Result Table** view. The **Result Table** view contains the following columns:

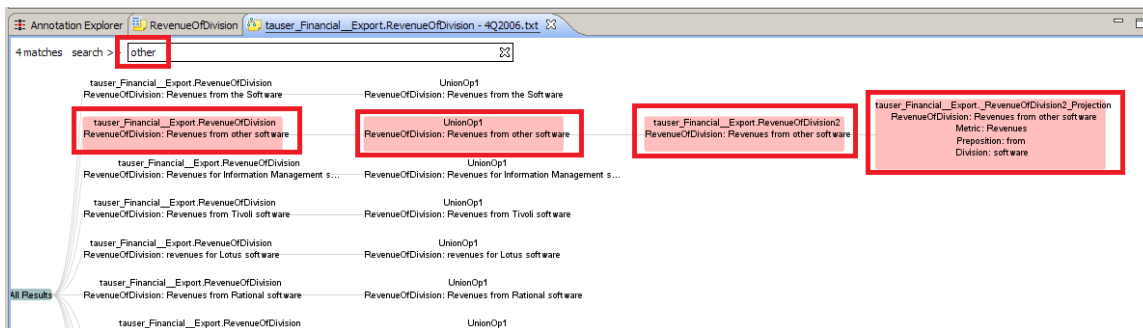
- One column for each attribute of the view
- One column indicating the document where a tuples was extracted from
- One **Explain** button



RevenueOfDivision (SPAN)	Input Document	Double-click this column to explain a tuple !
Revenues from the Software [2476-2502]	4Q2006.txt	Explain
Revenues from other software [2929-2957]	4Q2006.txt	Explain
Revenues for Information Management software [3296-3340]	4Q2006.txt	Explain
Revenues from Tivoli software [3421-3450]	4Q2006.txt	Explain
revenues for Lotus software [3597-3624]	4Q2006.txt	Explain
Revenues from Rational software [3770-3801]	4Q2006.txt	Explain
revenues from Global Technology Services [3972-4012]	4Q2006.txt	Explain

If we look carefully at this view, and in particular, focus on the rows highlighted, we see some mistakes: revenues associated with an incorrect division are output. For example, “other software” and “Information Management software” are not actually top-level divisions of IBM, they are actually subdivisions of the Software division.

2. You wish to understand the cause of these mistakes. Double-Click the **Explain** column of the second tuple. The **Provenance Viewer** is displayed. The Provenance Viewer displays the lineage of all RevenueOfDivision tuples extracted from a particular document (in our case 4Q2006.txt).



The lineage is essentially a graph of dependencies between intermediate tuples involved in generating a particular output. Each node is tagged with a view name and the actual data of the intermediate tuples in that view that is directly or indirectly responsible of generating an output.

3. To explain the second tuple (Revenues from other software) , we must first locate it among all RevenueOfDivision tuples generated for 4Q2006.txt. Enter a suitable search term in the search bar at the top, for example, enter “other”. Nodes that contain the word “other” become highlighted in light red (see the previous screenshot).

#### Hint: Navigating the Provenance View

*Other ways to navigate the Provenance Viewer include CTRL+Mouse move → moves the graph within the canvas, or CTRL+Mouse scroll → zoom in/out of the canvas.*

4. Click on the nodes circled above (**UnionOp1** and **RevenueOfDivision2**). Here **UnionOp0** means the first union operand of the view **RevenueOfDivision**, **UnionOp1** would be the second union operand and so on. The chain of links highlighted represents the lineage for the output (“Revenue from other software”). We can see that it is being generated by a RevenueOfDivision2 tuple, which in turn is generated from a revenueOfDivision2\_Projection tuple. So the provenance indicates the problem lies in the view RevenueOfDivision2\_Projection.

#### Hint: Jump to AQL Editor

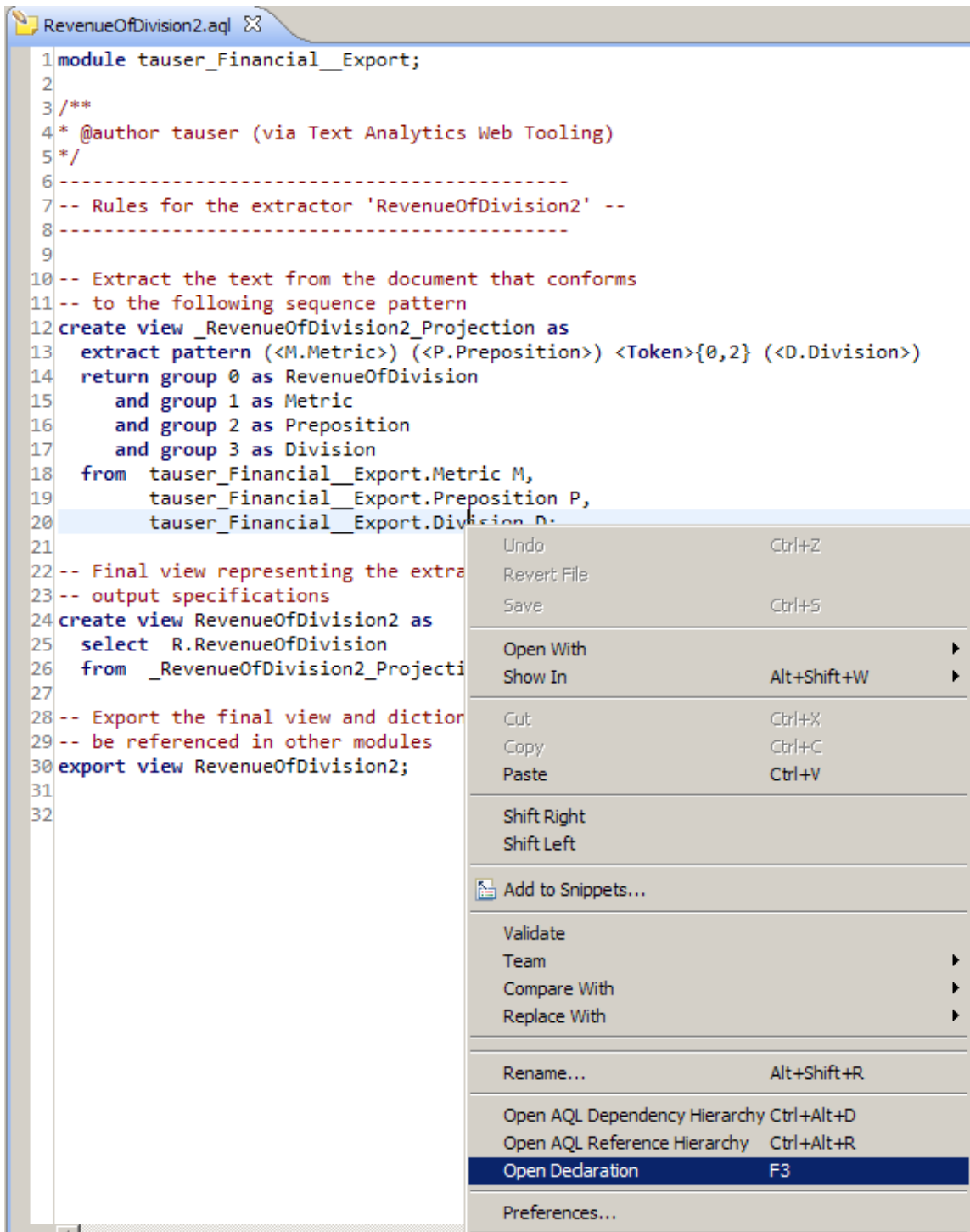
*In the Provenance Viewer, double-click on the node RevenueByDivision2\_Projection to jump to the definition of the view in the AQL editor.*

The definition of RevenueOfDivision2\_Projection is as follows:

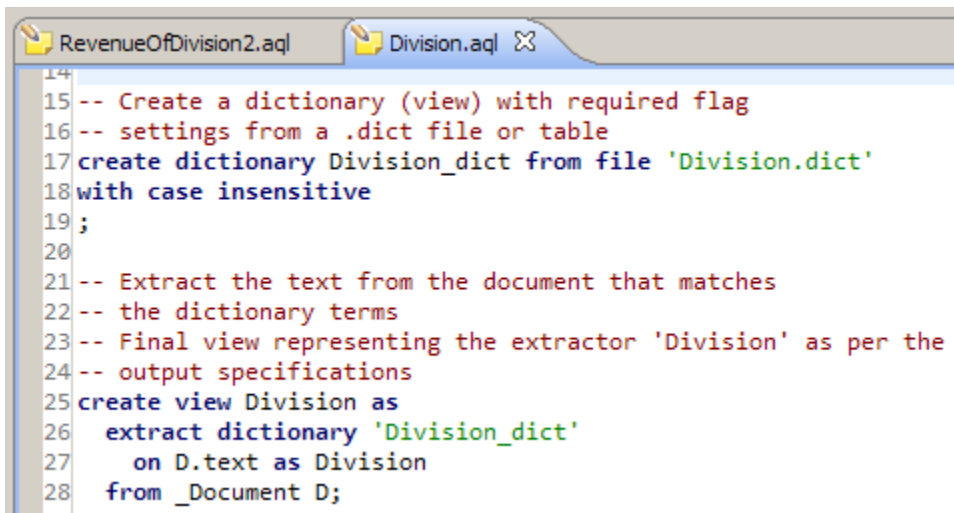
```
create view _RevenueOfDivision2_Projection as
  extract pattern (<M.Metric>) (<P.Preposition>) <Token>{0,2} (<D.Division>)
  return group 0 as RevenueOfDivision
    and group 1 as Metric
    and group 2 as Preposition
    and group 3 as Division
  from tauser_Financial__Export.Metric M,
       tauser_Financial__Export.Preposition P,
       tauser_Financial__Export.Division D;
```

Observe that RevenueOfDivision2\_Projection cannot be expanded further in the Provenance View. This is because RevenueOfDivision2\_Projection is defined using an “EXTRACT PATTERN” AQL statement, and this statement is not provenance-enabled yet. So to debug this statement, we need to look at its definition, and explore the lineage of its input views (Division, Metric and Revenue) separately. In this case, the views are fairly simple. We can quickly determine that there should be no problem with Metric, which simply identifies “revenues”, and with Preposition, which simply identifies “from”. But Division incorrectly identifies “software” (all lower case) as division name.

5. In the AQL editor, jump to the definition of Division by positioning the cursor on Division, right click and select Open Declaration (F3).



The AQL definition of Division opens in the AQL editor:



```

14
15 -- Create a dictionary (view) with required flag
16 -- settings from a .dict file or table
17 create dictionary Division_dict from file 'Division.dict'
18 with case insensitive
19 ;
20
21 -- Extract the text from the document that matches
22 -- the dictionary terms
23 -- Final view representing the extractor 'Division' as per the
24 -- output specifications
25 create view Division as
26   extract dictionary 'Division_dict'
27   on D.text as Division
28   from _Document D;

```

6. One way to avoid this mistake is to refine Division to add an additional predicate that requires the division name to start with a capitalized letter as follows:

```

create view Division as
  extract dictionary 'Division_dict'
    on D.text as Division
  from _Document D
  having MatchesRegex(/(\p{Lu}\p{M})*.*/ , Division);

```

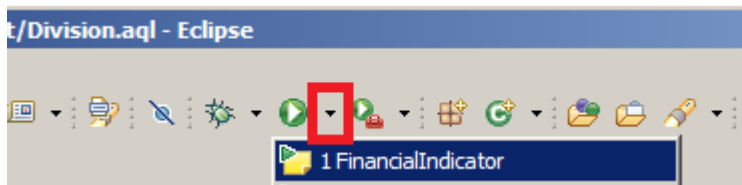
Here `(\p{Lu}\p{M})*.*` is a regular expression that matches any Unicode uppercase letter (`\p{Lu}`) followed by 0 or more graphemes (`\p{M}*`) followed by zero or more any characters.

7. Once you have made this change, save the file using **File > Save** or CTRL+S.

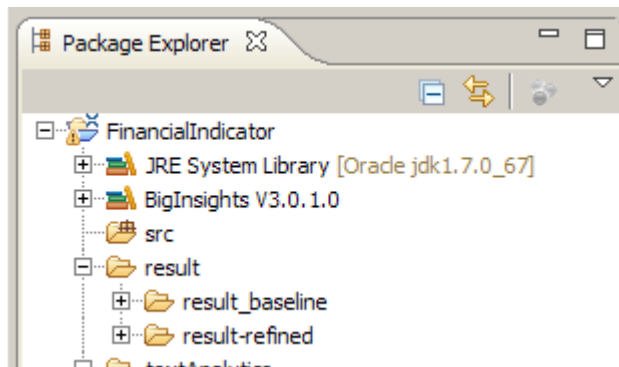
### Step 6: Using the Annotation Difference tool

You are ready to execute your refined extractor, and compare its results to the results from before, using the **Annotation Difference tool**. First, execute the extractor.

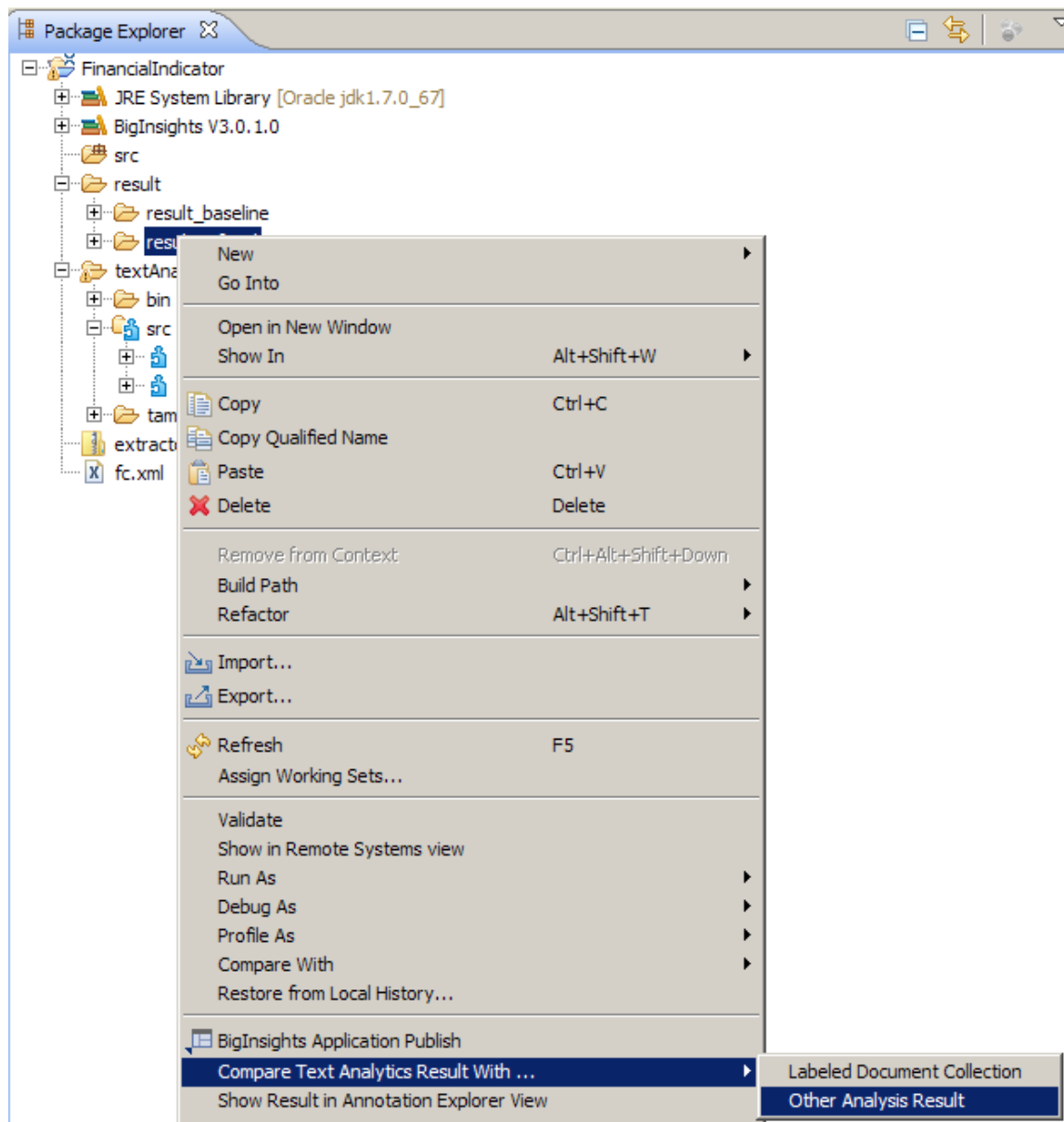
1. In the top level Eclipse Tool bar, click on the small arrow beside the **Run** button and select the **FinancialIndicator** configuration you created before:



2. In the Package Explorer, rename the most recent result folder to “result\_refined” as you did before. You should now see the following in the Package Explorer:

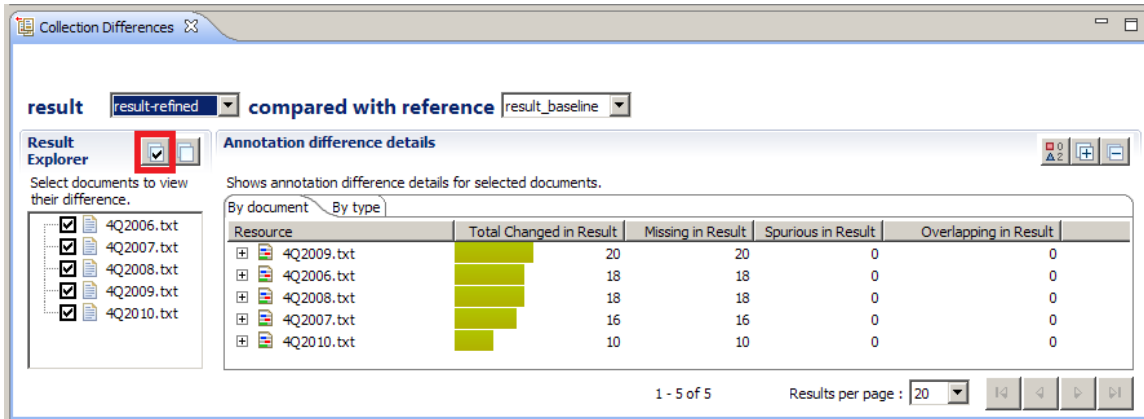


3. In the Package Explorer, right click “result\_refined” and select **Compare Text Analytics Result With ... > Other Analysis Result**, as shown below, then select the “result\_baseline” and click **OK**:



4. The results appear in the **Collection Difference** view. Click on the highlighted button to enable differences for

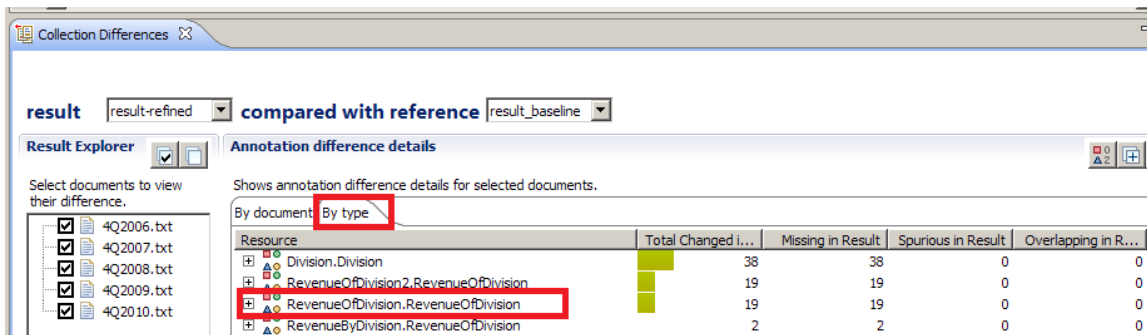
all documents.



Here you can see that the latest result “result\_refined” is compared with the reference result “result\_baseline”, and the count of differences is shown in the “By Document” view. The differences can be of three types:

- **Missing in result:** annotations present in the reference, but not the result
- **Spurious in result:** annotations present in the result, but not the reference
- **Overlapping in result:** annotations present in both the result and reference, but with slightly different offsets.

The “By document” view aggregates these results by document. Switch to the “By type” view to aggregate the results by type. Here, a type is a pair <ViewName>.<AttributeName>.

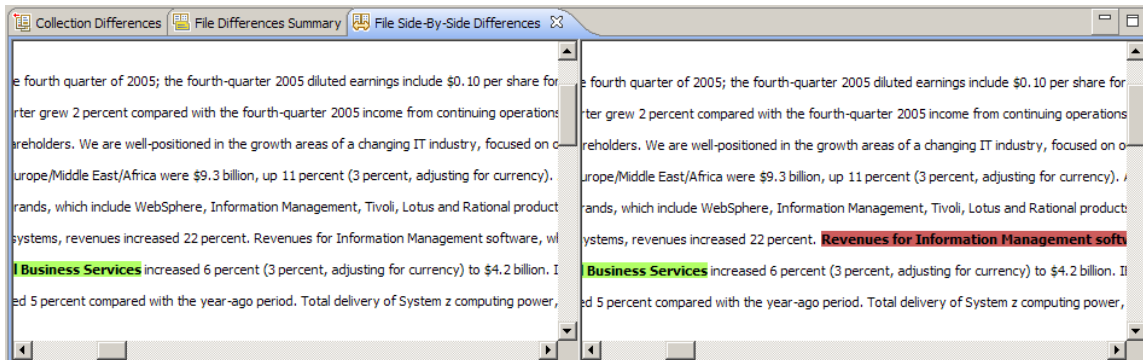


Observe how the only differences are related to the change you just made to Division, which directly impacts RevenueOfDivision2 and RevenueofDivision.

5. From the aggregated view, you can drill down to actual changes in output. Click on RevenueOfDivision.RevenueOfDivision. The actual differences in RevenueOfDivision. RevenueOfDivision annotations, for all input documents, are shown in the File Differences Summary view. Each annotation is color coded to indicate the type of difference (missing, spurious, overlapping or unchanged). You can see that some divisions remained unchanged (green). Incorrect divisions such as “software” have been removed from the refined result. Overall, your result looks much better!

Collection Differences			
File Differences Summary			
Input File Path	FinancialIndicator\result\result-refined	FinancialIndicator\result\result_baseline	Type
4Q2006.txt	Revenues from the Software [2476-2502]	Revenues from the Software [2476-2502]	Unchanged in Result
4Q2006.txt	revenues from Global Technology Services [3972-4012]	revenues from Global Technology Services [3972-4012]	Unchanged in Result
4Q2006.txt	revenues from Global Business Services [4098-4136]	revenues from Global Business Services [4098-4136]	Unchanged in Result
4Q2006.txt	Software segment revenues [9797-9822]	Software segment revenues [9797-9822]	Unchanged in Result
4Q2006.txt	Revenues from the Global Technology Services [9916-9960]	Revenues from the Global Technology Services [9916-9960]	Unchanged in Result
4Q2006.txt	Revenues from the Global Business Services [10073-10115]	Revenues from the Global Business Services [10073-10115]	Unchanged in Result
4Q2006.txt		Revenues from other software [2929-2957]	Missing in Result
4Q2006.txt		Revenues for Information Management software [3296-3340]	Missing in Result
4Q2006.txt		Revenues from Tivoli software [3421-3450]	Missing in Result
4Q2006.txt		revenues for Lotus software [3597-3624]	Missing in Result
4Q2006.txt		Revenues from Rational software [3770-3801]	Missing in Result
4Q2007.txt	Global Technology Services segment revenues [2571-2614]	Global Technology Services segment revenues [2571-2614]	Unchanged in Result
4Q2007.txt	Global Business Services segment revenues [2763-2804]	Global Business Services segment revenues [2763-2804]	Unchanged in Result
4Q2007.txt	Revenues from the Software [4237-4263]	Revenues from the Software [4237-4263]	Unchanged in Result
4Q2007.txt	Revenues from the Global Technology Services [8332-8376]	Revenues from the Global Technology Services [8332-8376]	Unchanged in Result
4Q2007.txt	Revenues from the Global Business Services [8490-8532]	Revenues from the Global Business Services [8490-8532]	Unchanged in Result
4Q2007.txt	Software segment revenues [8733-8758]	Software segment revenues [8733-8758]	Unchanged in Result
4Q2007.txt		Revenues from Information Management software [4098-4136]	Missing in Result

6. Finally, click on any row to drill down to the actual text of the document and see your annotations highlighted in the **File Side By Side Differences** view. The reference always appears in the right, whereas the result that is being compared with the reference appears in the left.



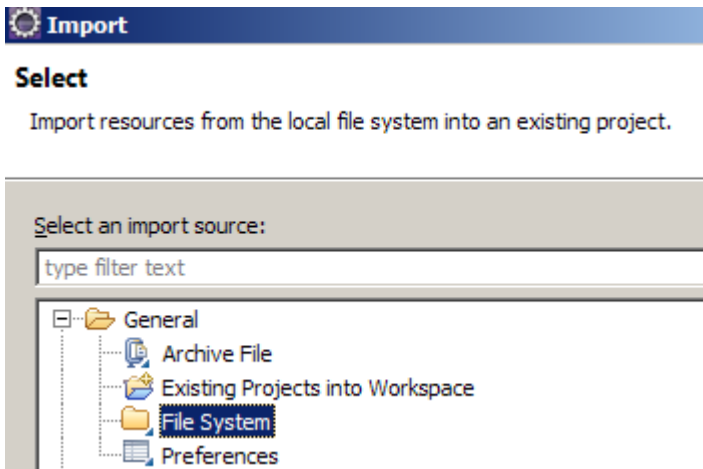
Note that the incorrect division “Information Management software” is no longer identified.

7. At this point, you can continue to evaluate the extractor manually, by examining the result, or you can choose to evaluate it in a more formal fashion using the **Labeled Collections** facility in combination with the **Annotation Difference Tool**.

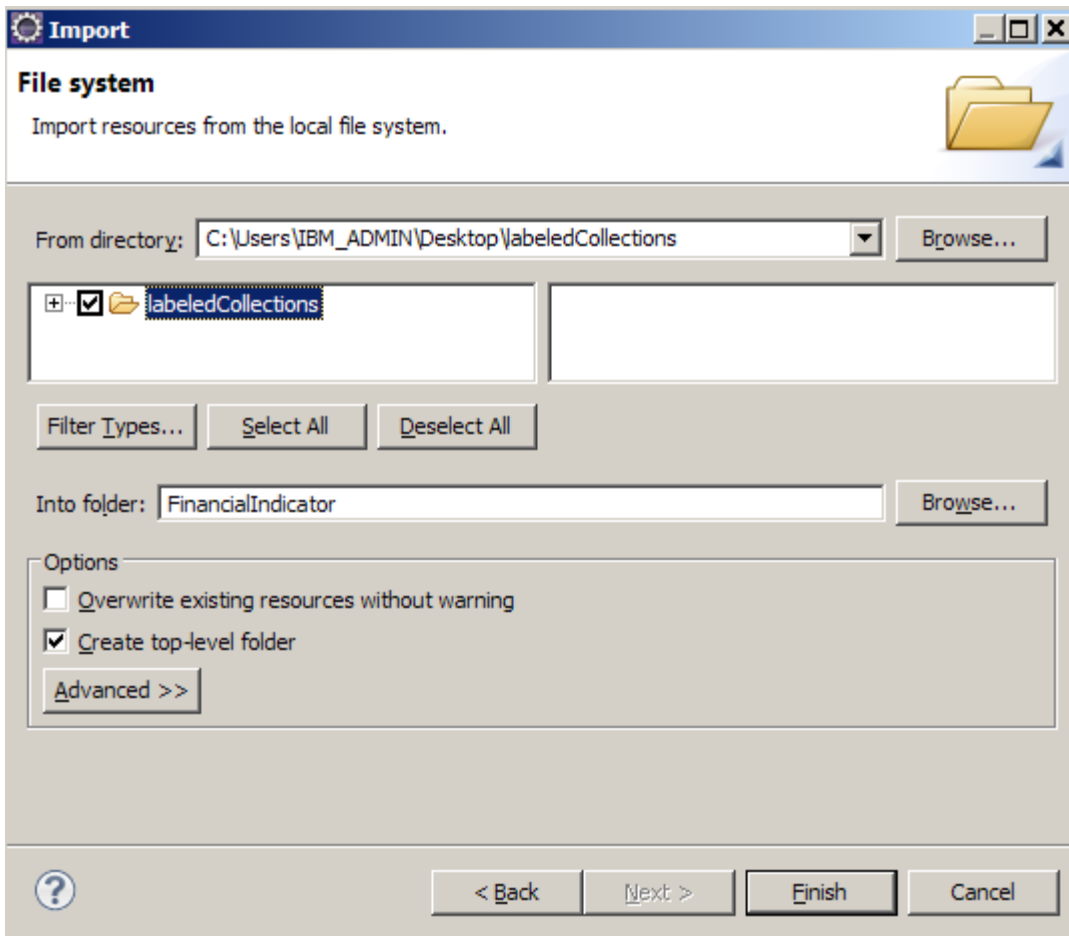
### Step 7: Using the Labeled Collection tool

A Labeled Collection is essentially a gold standard of results: it contains all correct mentions, and nothing else. Therefore, it can be used to formally evaluate an extractor by calculating precision, recall and F-measure of the extractor with respect to the gold standard.

1. A labeled collection has been provided to you in the lab materials in the folder labeledCollections/financialStatements. Import it in Eclipse by selecting **File > Import > General > File System**

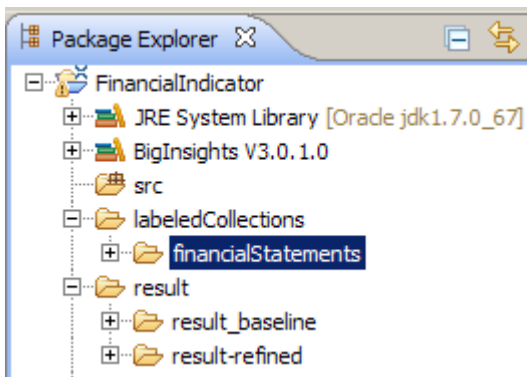


2. Fill in the Import dialog as follows, navigating to the location of the **labeledCollection** folder (in the directory where you have unpacked the supporting lab materials), then click **Finish**.



The labeledCollections folder should now be visible in Package Explorer:

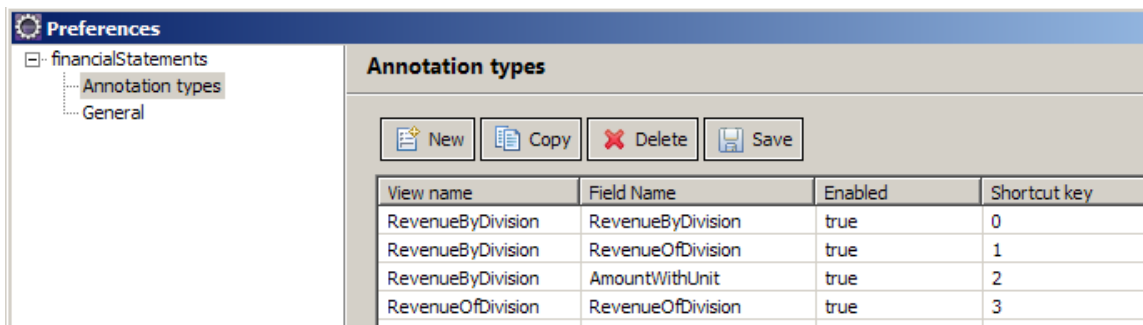




### Hint: Create your own Labeled Collection

You can create your own labeled collection by following instructions in the Eclipse Help. To access the Eclipse Help, go to **Help > Help Contents > BigInsights > Analyzing Text in Big Data > Comparing Results**.

3. Now, let's first examine the labeled collection. In the Package Explorer, from the context menu of your directory `FinancialIndicators/labeledCollections/financialStatements`, select: **Labeled Document Collection > Configure**. The Preferences dialog is displayed (you might need to expand the entry in the left panel).

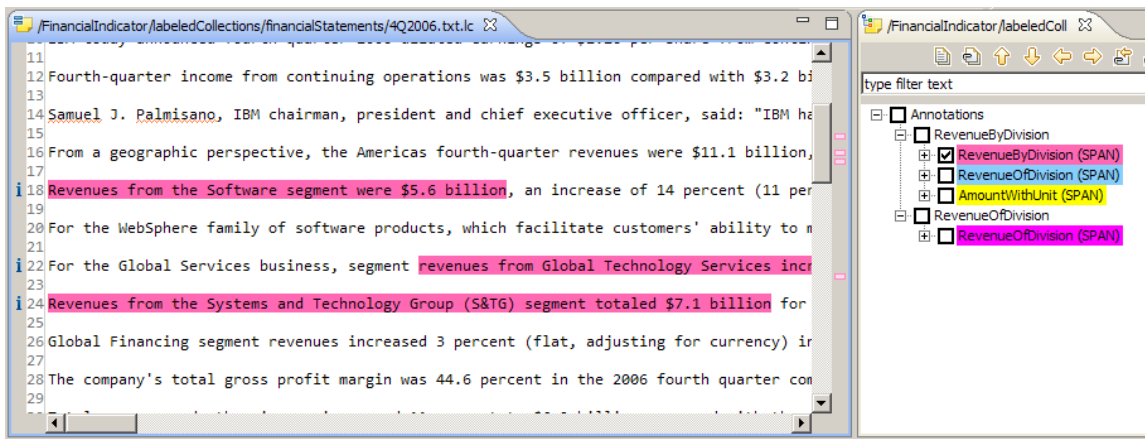


You can see how all annotation types (`viewName.attributeName`) in the refined result are imported. Each of them has an assigned shortcut that can be used to quickly annotate more mentions of a type in the document. For example, to annotate `RevenueOfDivision.AmountWithUnit` you would use **CTRL+2**

### Limitations

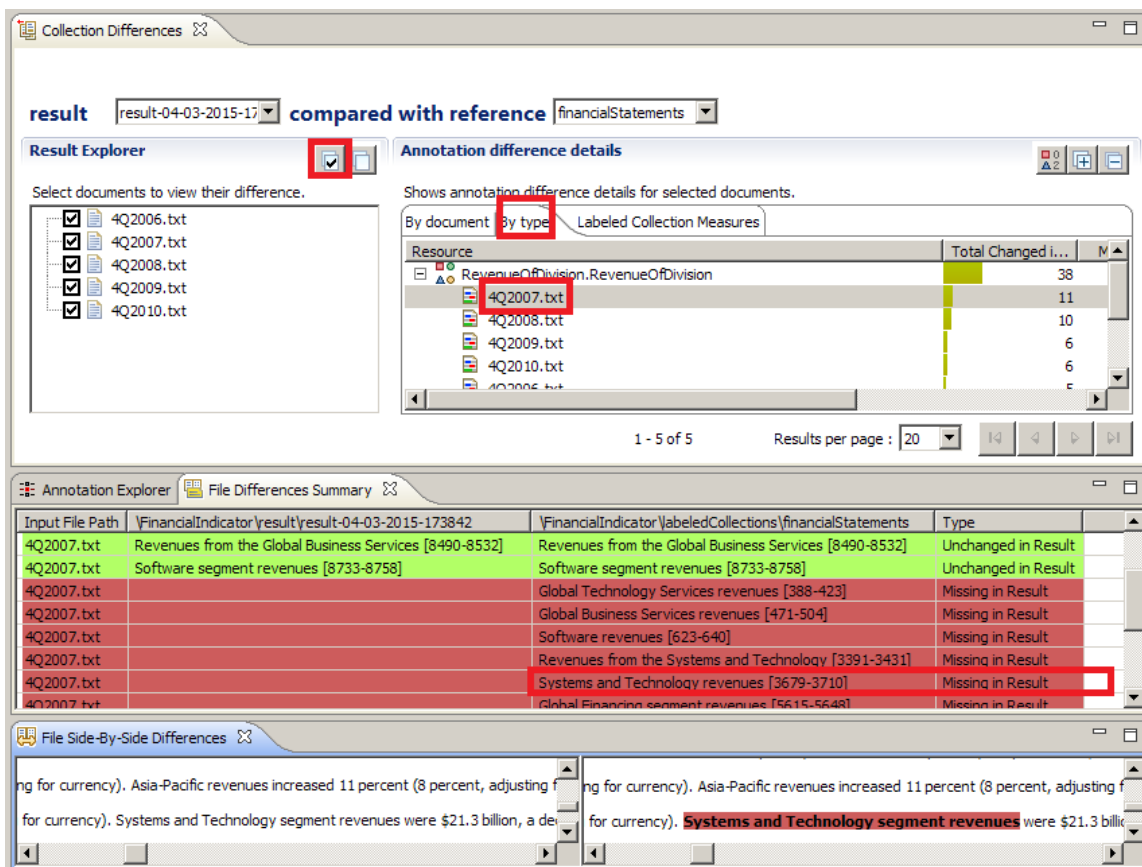
- Only annotations of type `Span over Document.text` are currently supported. The tool will throw an error if annotations of a different type are present in the result you import from.
- Only 10 annotation types can be assigned a shortcut (**CTRL+** number from 0 to 9)

4. In the **Package Explorer**, double-click the file `labeledCollections/financialStatements/4Q2006.txt.lc` to see some of the annotations that were imported. The labeled document opens in the editor, and you can select which annotation types to look at in the **Result Tree** view (right)



5. We now show you how to use the labeled collection in evaluating the quality of an extractor. Rerun the extractor on the entire document collection. Then, in the Package Explorer, right click the most recent result and select **Compare Text Analytics result with > Labeled Document Collection**. Choose **financialStatements** from the dialog. The results appear in the Comparison View which we have seen earlier when we discussed the **Annotation Difference tool**. Essentially, everything in this view is as described in **Step 6**. This means that you can click on a type to drill down from aggregated results to fine grained results and see how they differ from the labeled collection.

6. Try it yourself, clicking on the buttons circled in red:



For example, here we can see that the extracted result misses some annotations, for example, "Revenues from the Systems and technology" and "Systems and Technology revenues".

**EXERCISE 1:**

Which part of your extractor you should modify in order to capture this missing annotations? **HINT:** recall the dictionary of division names.

The main difference when comparing a result with a labeled collection, as opposed to another result, is the section **Labeled Collection Measure**, which gives you a measure of the precision (accuracy), recall (coverage) and overall F-measure (combined measure of precision and recall) for each annotation type, where:

- **Precision:** percentage of correct results among all results. Measure of accuracy of your extractor
- **Recall:** percentage of correct results identified, among all true mentions. Measure of coverage
- **F-measure:** combined measure of precision and recall

•

Resource	Precision	Recall	F-Measure
RevenueOfDivision.RevenueOfDivision			
RevenueByDivision.RevenueOfDivision			
RevenueByDivision.AmountWithUnit			
RevenueByDivision.RevenueByDivision	61.22	66.67	63.83
Exact	61.22	66.67	63.83
Partial	63.71	69.38	66.43
Relaxed	71.43	77.78	74.47

There are three flavors for the measures, depending on how we count partial (overlapping) mentions:

- **Exact:** partial matches are considered incorrect (counted as 0)
- **Relaxed:** partial matches are considered correct (counted as 1)
- **Partial:** partial matches are partially accounted for based on the percentage of overlap.

For example: if the labeled collection contains “Anna Magnani” but the extractor identifies only “Anna”, we consider the result incorrect for Exact measure, completely correct for Relaxed measure and partially correct for Partial measure, with score 0.38 (since the overlap is 5 characters out of 13). In our case, we don’t have any partial overlaps, so all measures are equal.

The three measures give you an idea of the improvement you can achieve given that you fix the rules such that all partial matches are identified completely.

**EXERCISE 2:**

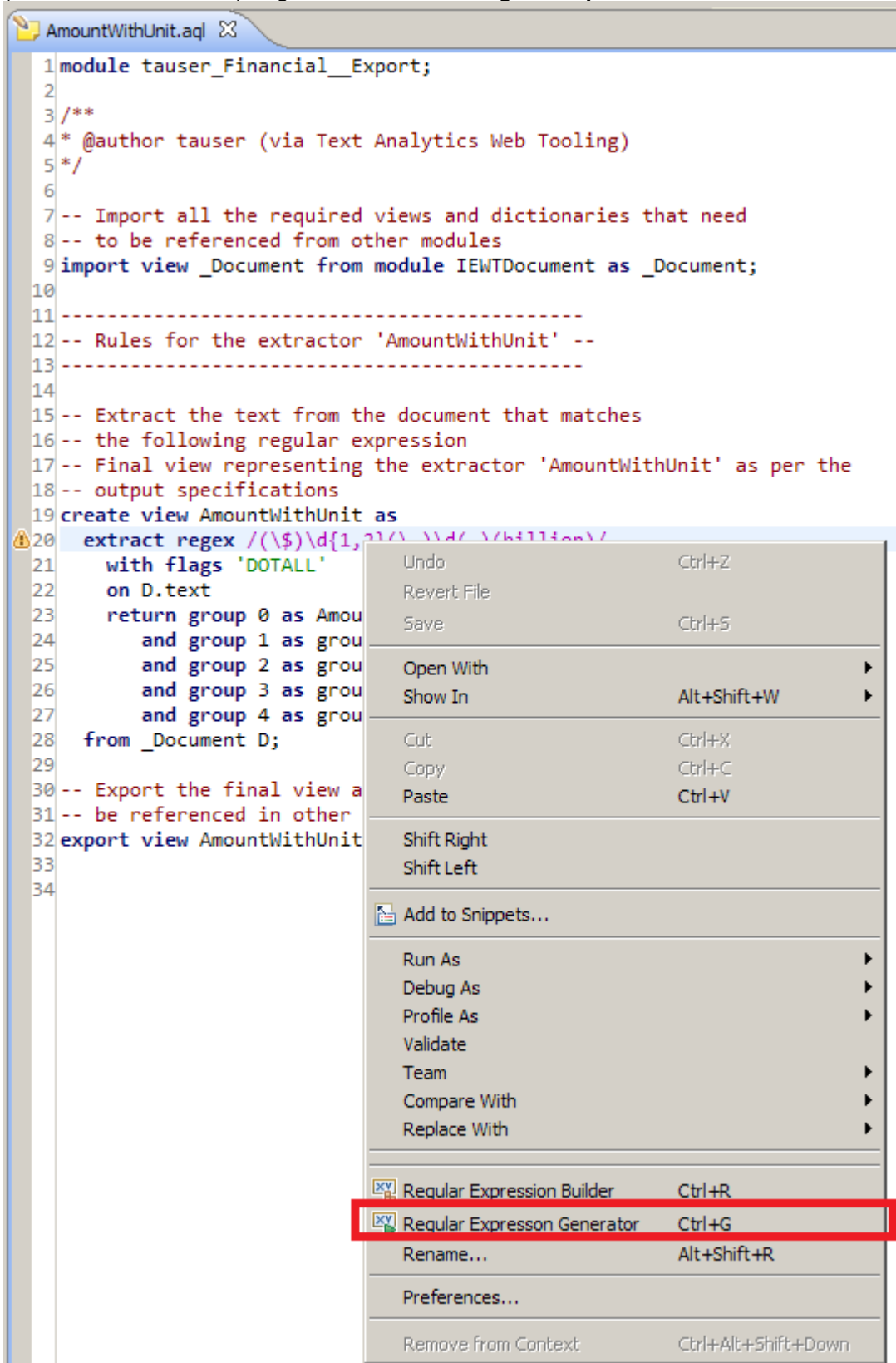
We can see that the accuracy and coverage of RevenueByDivision.RevenueByDivision are not satisfactory (61.22% and 66.67%). Start examining the differences between the extraction results and the labeled collection. Then refine your extractor to achieve higher coverage.

**ANSWER KEY:** The solution is in the project FinancialIndicatorSolution.zip. You can import this project in Eclipse following the procedure in Step 1 (Backup).

**Step 8: Using the Regular Expression generator**

Remember when you have created the AmountWithUnit concept in the Web Tooling and have specified the regular expression as `/(\$)d{1,2}(\.)(d( )billion)/`. This tool helps you generate such a regular expression from a few examples.

1. To start the Regular Expression Generator, position your cursor in the AQL Editor on a regular expression (between the // markers). Right click and select **Regular Expression Generator**.



2. In the **Samples** section, type some examples of text that you would like the regular expression to capture.

3. Select **Generate Expression**, then select the first expression and click **Next**:

**Regular Expression Generator**

Begin by directly adding samples to the table or loading them from a file. Click on the Generate regular expression button. Once the expressions are generated,

Load samples from file

Samples
\$5.6 billion
\$7.3 billion

Generate regular expression

Regular Expressions

Each of the regular expressions below matches all strings of the sample set. Select one and click Next to refine it.

Specific regular expressions 
|
|
|
|
|
 General regular expressions

☒ \(\$\)d(\. )d( )(billion)  
 Try this regular expression if the concept does not have many optional parts.

☐ \(\$\)d(\. )d( )(b)(i)(l)(i)(o)(n)  
 Try this regular expression if the concept has many optional parts.

? < Back **Next >** Finish Cancel

4. The second screen of the **Regular Expression Generator** shows you how the generated expression matches various parts of the examples. As you understand how various parts of the regular expression match the examples, you can also think how to generalize the expression even further. For example, you might notice that the `\d` construct matches a single digit. You could relax this constraint by modifying the fields “This sub-expression contains at most ... characters” from 1 to 2 and click **Apply**.

**Regular Expression Generator**

Refine the regular expression by entering detailed information for each subexpression. The subexpression you are currently working on is highlighted.

Refine the current subexpression:

Subexpression: `\d`

☒ Any of the symbols in this character class:

Character Class: **Digits** Change

This character class comprises any digits (`\d`).

This subexpression contains at least  characters.

This subexpression contains at most  characters.

☐ An integer number within a certain range:

Minimum (of the number range):

Maximum (of the number range):

☐ Allow leading zeros

☐ Alternation of the following Samples:

Samples of this subexpression

5
7

Click at a row of this table to add or remove samples.

☐ This subexpression is optional. Apply

**Regular Expression:**

`((\d)\d(\d)(billion))`

<< Back Next >>

Match	Samples
✓	\$5.6 billion
✓	\$7.3 billion

Import...

Type some text to test your regular expression.

Back Next > Finish Cancel

Notice how in the regular expression, the first `\d` has been changed to `\d{1,2}`.

5. You can repeat the previous step on your own, to refine the quantifier for the second `\d` element of the regular expression. Therefore, your regular expression will now capture amounts with both sides of the decimal point consisting of 1 to 2 digits (as opposed to a single digit). You can test this is the case by typing some sample text, for example, \$56.73 billion in the bottom right corner of the dialog. You will notice that the sample text becomes highlighted, a sign that it is now captured by the refined regular expression.

**Regular Expression Generator**

Refine the regular expression by entering detailed information for each subexpression. The subexpression you are currently working on is highlighted.

Refine the current subexpression:

Subexpression: **(billion)**

☐ Any of the symbols in this character class:

Character Class: **Lower-case Letters** Change

This character class comprises lower case letters (p{L}).

This subexpression contains at least:  characters.

This subexpression contains at most:  characters.

☒ Alternation of the following Samples:

Samples of this subexpression:

Click at a row of this table to add or remove samples.

☐ This subexpression is optional. Apply

**Regular Expression:**

`((\d{1,2})(\d{1,2}))(billion)`

<< Back Next >>

Match	Samples
✓	\$5.6 billion
✓	\$7.3 billion

Import...

`$56.73 billion`

< Back Next > **Finish** Cancel

6. Once you are satisfied, click **Finish**. The refined regular expression is automatically inserted inside the `//` markers of the **AmountWithUnit** view, which now should look like this:

```
create view AmountWithUnit as
  extract regex /(\d{1,2})(\d{1,2})(billion)/
  with flags 'DOTALL'
  on D.text
  return group 0 as AmountWithUnit
  and group 1 as group_1
  and group 2 as group_2
  and group 3 as group_3
  and group 4 as group_4
from _Document D;
```

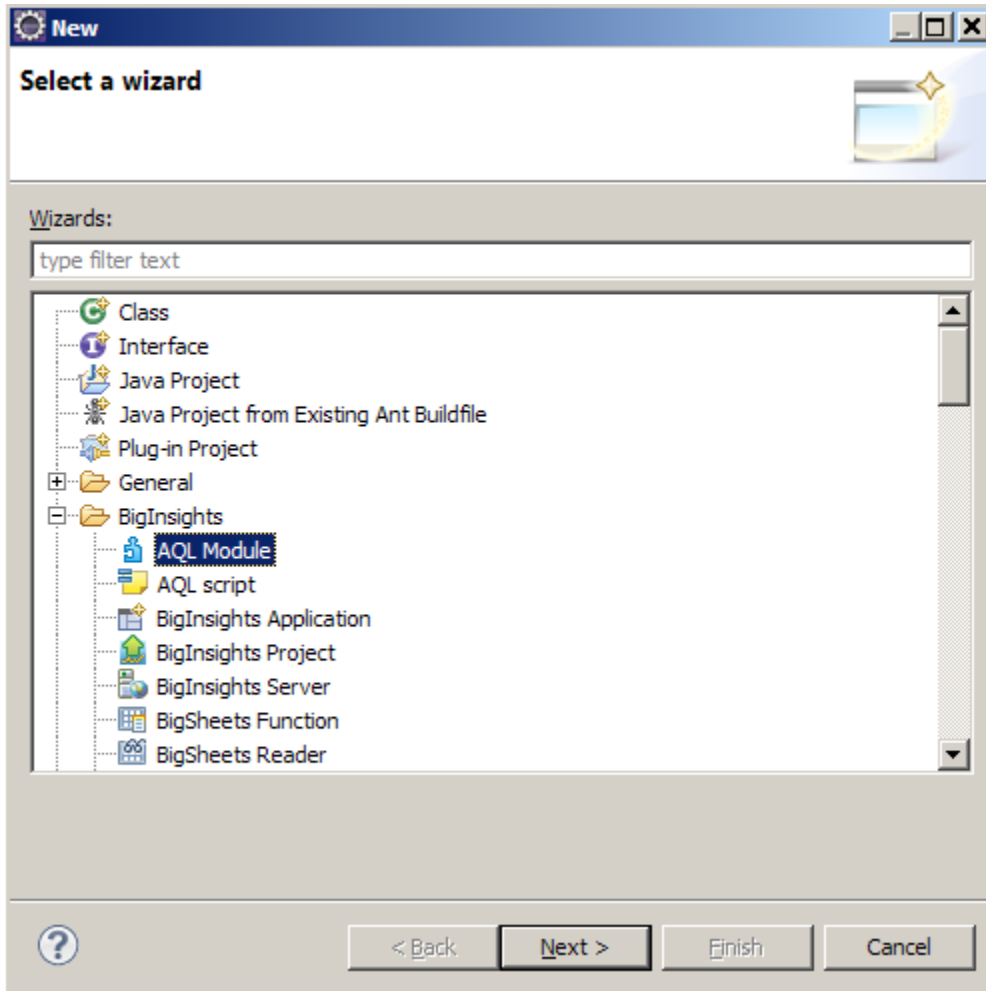
7. Save the **AmountWithUnit.aql** file to persist the changes.

## Step 9: Using the Pattern Discovery tool

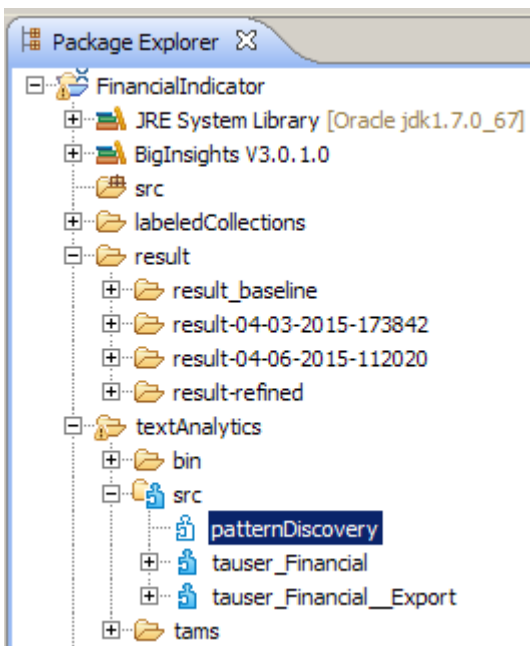
In the final part of this lab let's assume we want to extend our extractor to consider other financial metrics other than revenue. Since you might not have a good idea of all possible financial metrics in our domain, you can use the **Pattern Discovery** tool to discover such metrics. Once you do so, you can collect them in a dictionary of financial metrics, and use that dictionary to extend the rules.

1. The first step in doing Pattern Discovery is to specify the input. We do so by defining a view in AQL. Since this is exploratory code, we will define it in a separate AQL module, to maintain it separately from your main AQL code for our Revenue by Division work (i.e., the modules **tauser\_Financial\_\_Export** and **tauser\_Financial**).

2. Create a new AQL module from **File > New > Other > BigInsights > AQL Module**.

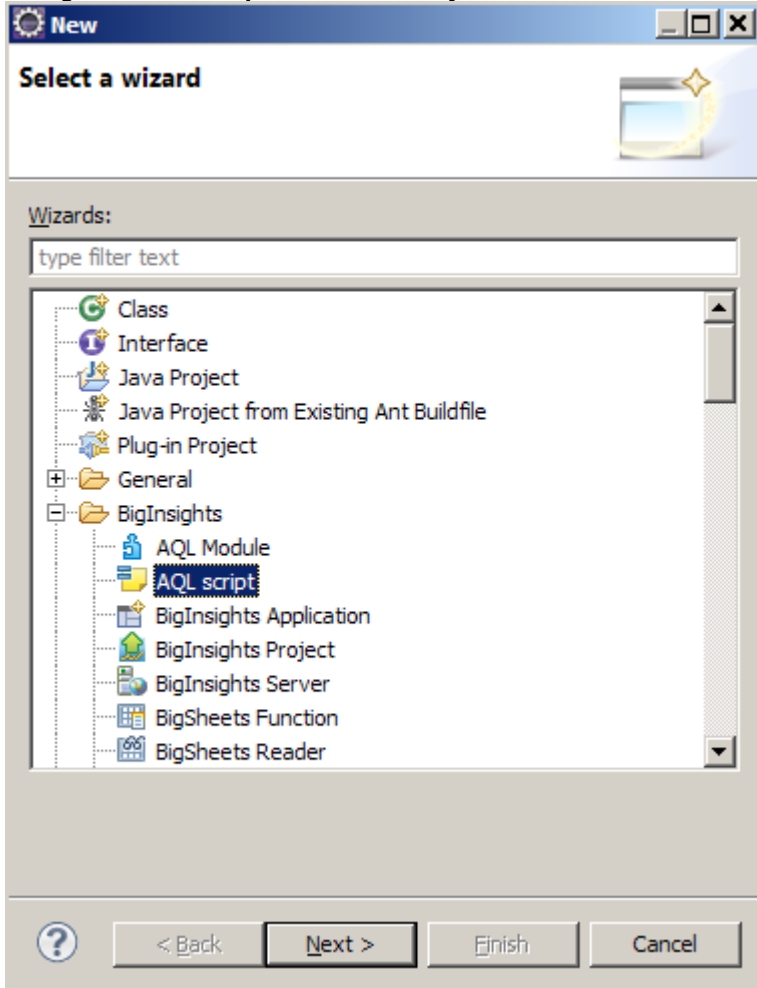


2. Type **patternDiscovery** as the module name and click **Finish**. The new module appears in Package Explorer:

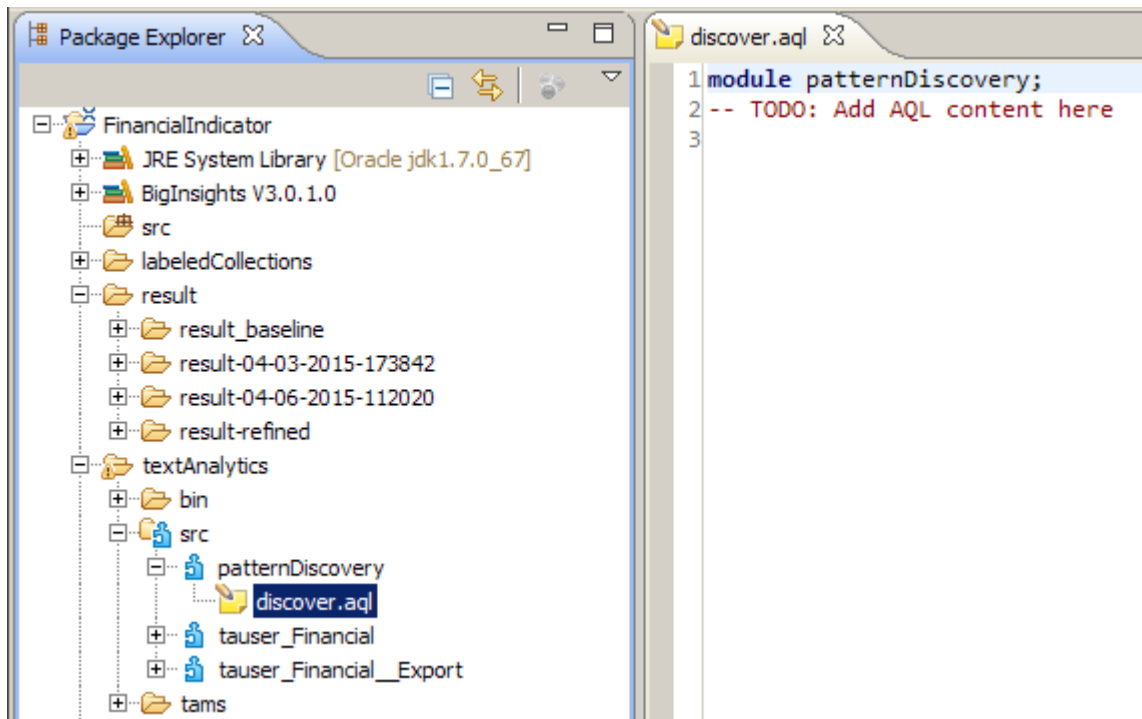




3. Right click on the **patternDiscovery** module and select **New > Other > BigInsights > AQL Script**.



4. Enter `discover.aql` as **File Name** and click **Finish**. The new AQL file `discover.aql` appear in the Package Explorer and opens in the AQL Editor.



5. You are now ready to define the input to Pattern Discovery tool. Since financial metrics usually occur before an amount, a reasonable idea might be to look for patterns in the vicinity of an amount. In particular, we will use a view that extracts 10 tokens to the left of an AmountWithUnit annotation.

Copy the following statements to discovery.aql.

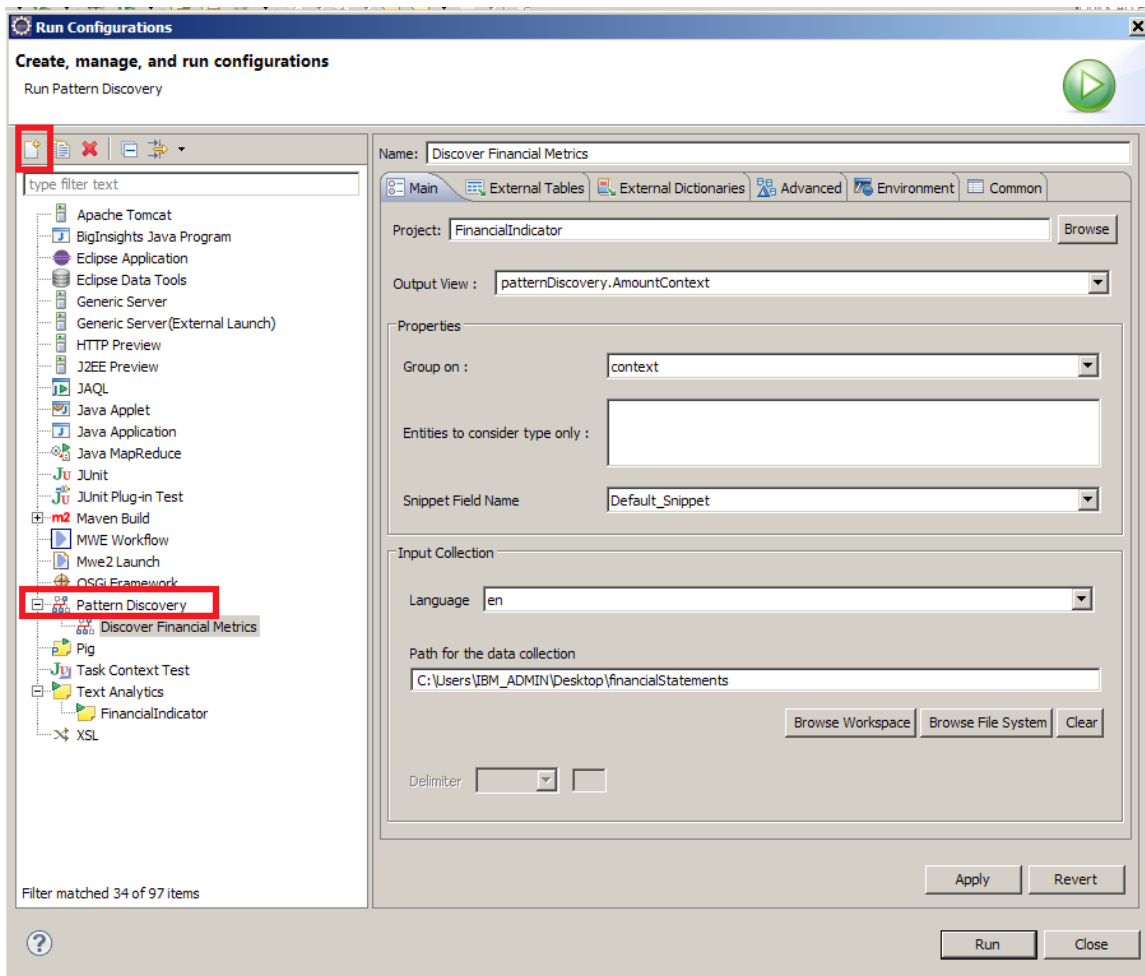
```
-- Import the necessary view from another module
import view AmountWithUnit from module tauser_Financial__Export as AmountWithUnit;

-- Extract the 10 tokens to the left of an amount
create view AmountContext as
select LeftContextTok(A.AmountWithUnit, 10) as context
from AmountWithUnit A;

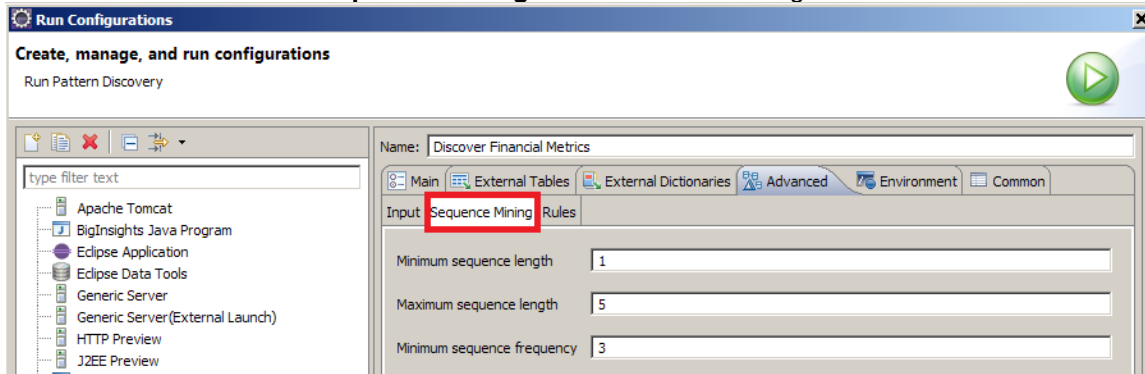
output view AmountContext;
```

6. Save discovery.aql

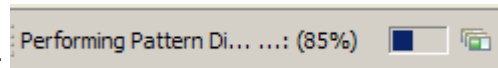
7. In the Eclipse menu, select **Run > Run Configurations > Pattern Discovery**. Create a new Pattern Discovery configuration by selecting the top left button, and fill in the dialog as follows:



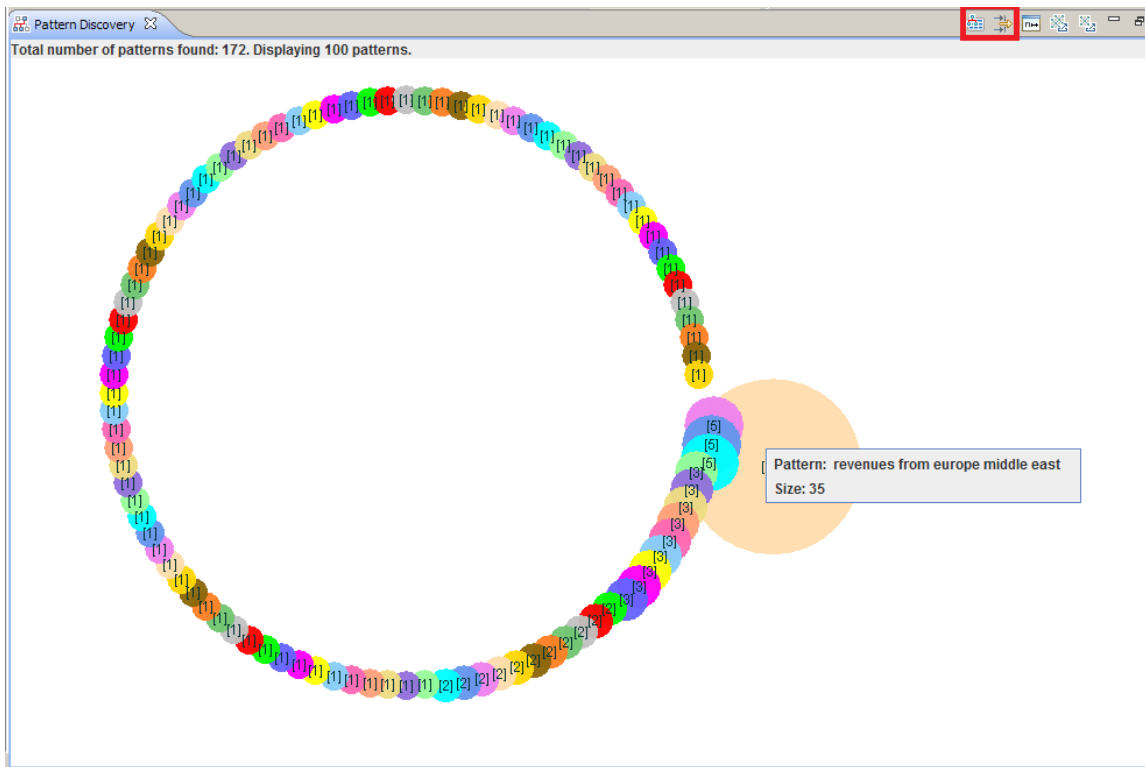
8. Go to the **Advanced > Sequence Mining** and enter the following values




9. Click **Finish** and Pattern Discovery will run in the background:



10. When completed, the results appear in the Pattern Discovery View:



#### Hint: Explore beyond the top 10 patterns

Only the top 10 most common patterns are shown initially. To view less frequent patterns, click the  icon in the Pattern Discovery view, and explore groups by size.

11. Furthermore, you can also see the patterns in a table, by selecting the button **Display Pattern Discovery Result in a Table**, also in the top right corner. You may need to click the **Filter** button again to see more than 10 patterns.

Pattern Discovery    Pattern Discovery Table

Total number of patterns found: 172. Displaying 100 patterns.

Id	Size	Pattern	Original Pattern
1	35	revenues from europe middle east	{revenues from europe middle east}
2	5	shares outstanding n debt including	{shares outstanding n debt including}
3	5	an ; management tivoli lotus and rational	{an;management tivoli lotus and rati...
4	5	adjusting for currency global financing	{adjusting for currency global financi...
5	3	to 1 non global financing	{to 1 non global financing}
6	3	perspective the americas full year	{perspective the americas full year}
7	3	perspective the americas fourth quarter	{perspective the americas fourth qu...
8	3	total expense and other	{n total expense and other}
9	3	generated free cash flow of	{generated free cash flow of}
10	3	from a management segment view	{from a management segment view}
11	3	for currency systems and technology	{for currency systems and technology}
12	3	adjusting for currency total services	{adjusting for currency total services}
13	3	3 percent adjusting for currency	{3 percent adjusting for currency}
14	2	year	{year}
15	2	percent	{percent}
16	2	percent ; 9 percent	{percent;9 percent}
17	2	percent ; 10 percent	{percent;10 percent}
18	2	percent 4 percent adjusting for	{percent 4 percent adjusting for}
19	2	net income was	{net income was}
20	2	cash flow of \$	{cash flow of \$}
21	2	8 ; percent adjusting for currency to	{8;percent adjusting for currency to}
22	2	5 percent adjusting for currency ; to	{5 percent adjusting for currency;to}

Observe that while the word “revenues” revenue” occurs very frequently, there are other types of financial metrics that are present, for example: *Shares outstanding, cash on hand, total expense, net income, cash flow*

12. Explore the table on your own and build a dictionary of financial metrics.

### EXERCISE 3:

Collect a dictionary of such metrics, then write an extractor for relationships between generalized Financial Metric and Amount, for example:

- Income from continuing operations of \$5 billion
- Loss from discontinued operations of \$100 million

The final output view of your extractor should be named Indicator and have three columns:

- **metric** (the span indicating the metric such as “Income from continuing operations”),
- **amount** (the span indicating the value of the metric such as “\$5 billion”)
- **match** (the span that covers both metric and amount)

**ANSWER KEY:** The solution is in the project FinancialIndicatorSolution.zip. You can import this project in Eclipse following the procedure in Step 1 (Backup).

**Congratulations!**  
**You have completed the SystemT Eclipse Tooling Lab!**