IBM Software
Information Management

Lab 04: Text Analytics

Hands-On Lab

Table of Contents

1. Introduction

InfoSphere™ BigInsights comes with sophisticated text analytics capabilities that allow users to easily specify rules to extract actionable insights from large amounts of text.

Annotation Query Language (AQL) is a language for building these "rules sets" or extractors that extract structured information from unstructured or semi-structured text. AQL is the primary method of creating new extractors in the InfoSphere BigInsights text analytics system.

In this lab we will analyze the press releases for IBM's quarterly earnings from the year 2006 to 2010 to extract the names of IBM divisions and their respective revenues.

2. Objectives

After completing this hands-on lab, you'll be able to:

- Explore and use the text analytics tooling environment.
- Run information extractors over a sample data set.
- Explore how to view, understand, and debug results from the extractor.

The Lab is organized in six sections following the six tasks shown in Figure 1.

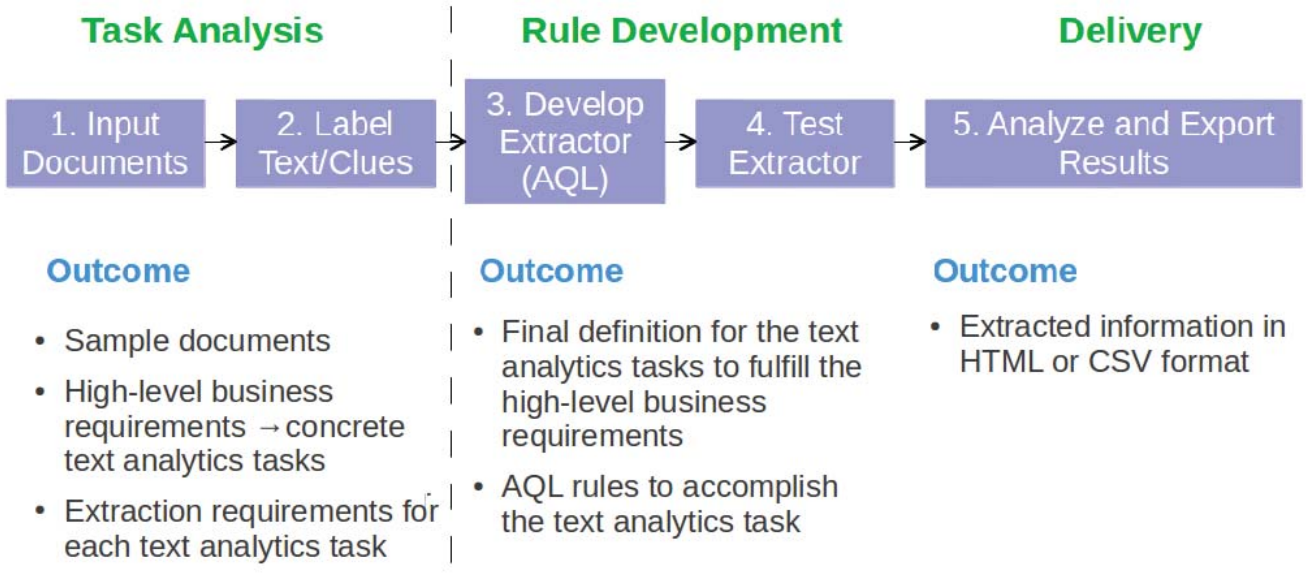

Figure 1 - Flow for best practices for deploying text analytics

Section 1: Importing input documents and dictionaries

This section guides you through the steps for setting up the extraction project in Eclipse IDE, importing the press release input files in the proper directory, and importing the dictionary of IBM division names in its proper location.

Section 2: Label Text / Clues

This section guides you through the steps involved in developing the 'Extraction Plan'. In this step we identify two concepts of interest and several basic features.

- First Concept: Reference to the revenue of a division
- Second Concept: The reported revenue for that division

Basic features are components of the above concepts, for example integers and decimal numbers are components of reported revenue. The occurrence of division names is a component of the first concept.

Section 3: Develop and Test Extractor (AQL)

In this section you go through the steps for writing the extraction logic (AQL code) for basic features. You will write extractors for three basic features: *division names, numbers* and reference to the term 'revenue'.

Section 4: Develop and Test Extractor (AQL) - continued

This part of the lab guides you through the steps for building upon the basic features of section 3 to extract the 'revenue of division' and the 'reported revenue' concepts in a single extracted unit.

Section 5: Analyze and Export the Results

In section 5 of the lab you will explore the tools for viewing the results from above test extraction tasks. The four tools are for:

1. Viewing an 'Output view in tabular form'.
2. Setting up filters for rows displayed in Annotation Explorer. They are covered in sections 5a and 5b respectively
3. Exporting all Output Views as an HTML and CSV files.
4. Viewing details of highlighted text in annotated document using mouse-over capability.

3. Importing input documents and dictionaries

Become acquainted with the BigInsights Text Analytics perspectives in the Eclipse tool, and the 'Extraction Task' and 'Extraction Plan' views in it.

- Become familiar with the steps for setting up Extraction projects.
- Importing input files and dictionaries.
- Examining the properties of the project and input files.

3.1 Creating the text analytics project in Eclipse

1. Login on to Virtual Machine using the following:
    a. Username = **biadmin**
    b. Password = **password**
2. On Desktop, Double click on Eclipse. And select the default workspace location provided.

Figure 2 - Launch Eclipse

3. Open the BigInsights perspective by clicking on the **Open perspective** button (top right corner) as shown in the figure below, and clicking on **BigInsights**

Figure 3 - Open Perspective

4. From the *Task Launcher for Big Data* select the **Tasks - Develop** as shown below. Alternatively, you can click on the **Develop** tab, also highlighted in the figure below.

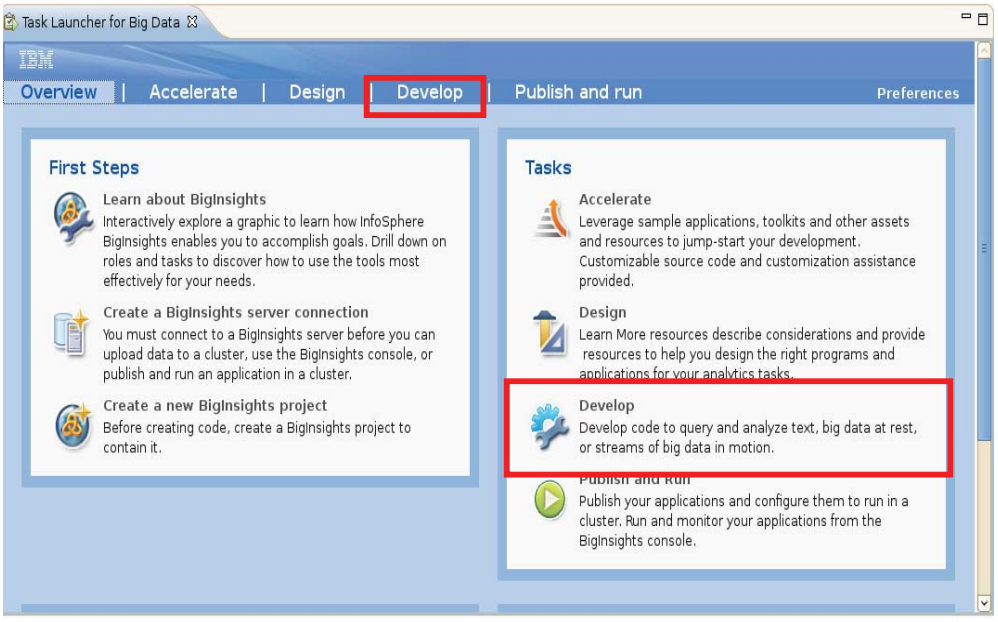

Figure 4 – Select Develop Tasks view in Eclipse

5. After navigating to the **Develop** tab of the *Task Launcher for Big Data,* select **'Create a text extractor'** from the tasks listed.

Figure 5 – Create a text extractor

6. Enter the project name '***MyFirstProject***' in the popup window and click **Finish**.

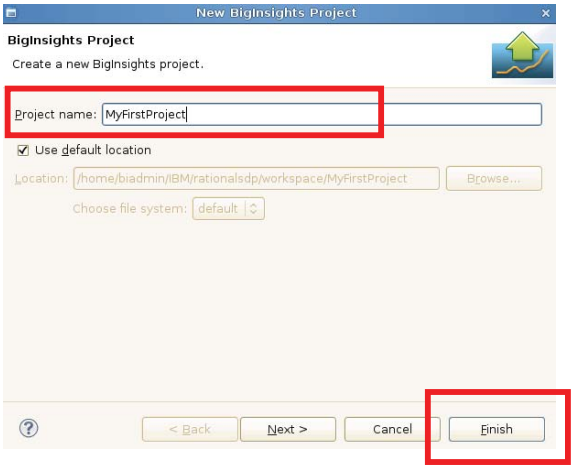

Figure 6 - Define the project name

Now you will be switched automatically to another perspective called *BigInsights Text Analytics Workflow*

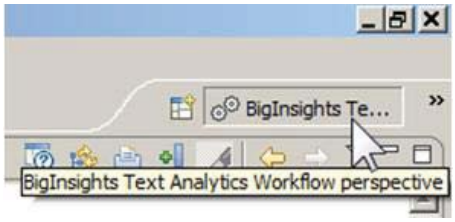

Figure 7 - BigInsights perspective button

> **Note: The Text Analytics Workflow perspective could also have been selected directly using the '*Open Perspective'* button on the top right corner of the IDE.**

Two key views are shown from this perspective:

- **Extraction Tasks** view, shown on the top left, consistent with the steps outlined in Figure 8.
- **Extraction Plan** shown on the right.

  Currently the plan shows the top level project name (*MyFirstProject*) and it is empty.

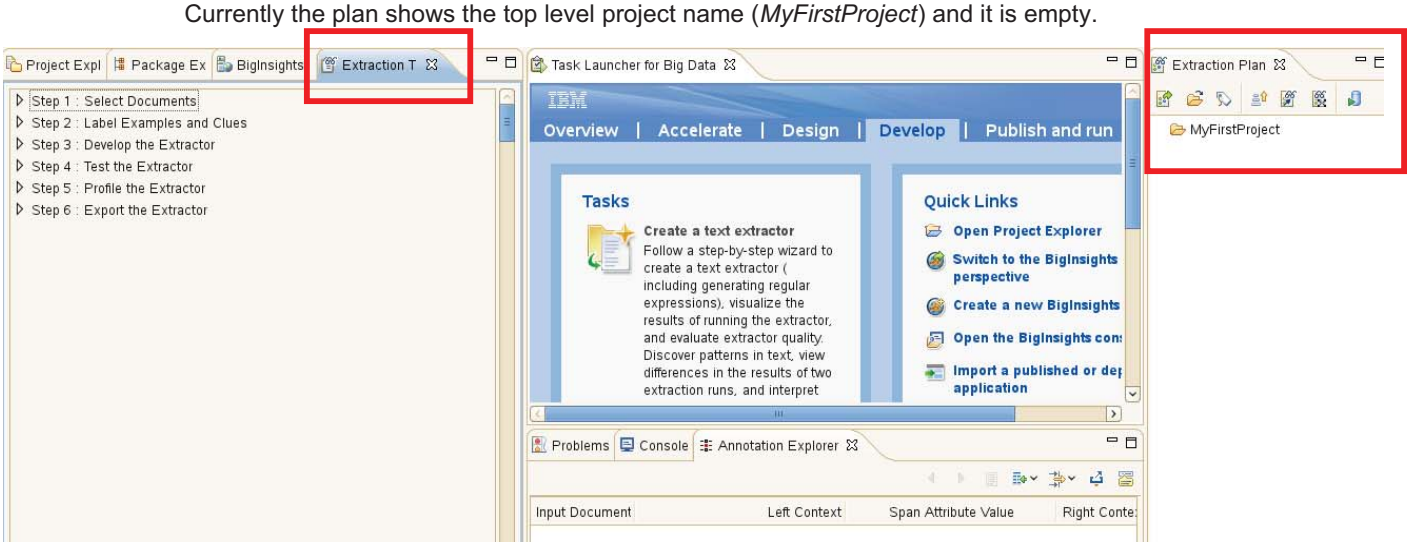

Figure 8 - BigInsights Text Analytics Perspective

> **Note: Extraction Tasks view can also be open using Window menu → Show View → Extraction Tasks**

A project structure is also created automatically. See the structure in the **Package Explorer** view, which is close to the *Extraction Tasks* view. If you cannot find it, you can also use the Eclipse menu, *Window -> Show View > Package Explorer*.

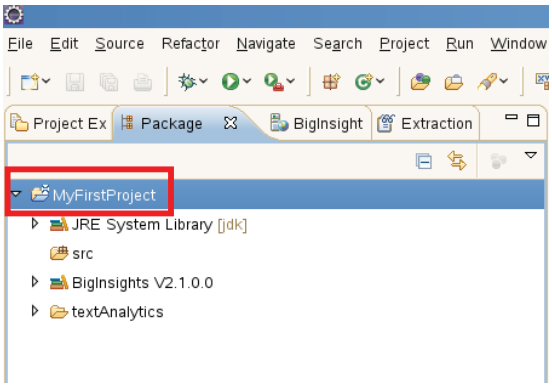

Figure 9 - The default package view in BigInsights Text Analytics perspective.

7. To verify the project properties right click **MyFirstProject** in the Package Explorer view and select **Properties**.

8. In the *Properties* dialog, expand **BigInsights** and then choose **Text Analytics**.

   Within this window, you can specify the source for this project or add associated projects for your current project.

9. Click on **Cancel** to close the properties dialog.

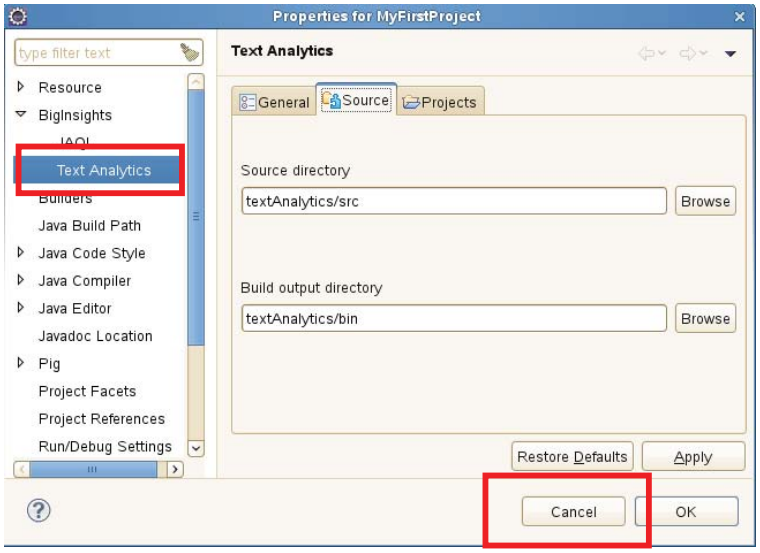

Figure 10 - Validating properties of a BigInsights Text Analytics project

3.2 Copying input files and dictionaries

1. Create a new folder named **data** under project **MyFirstProject** in the Package Explorer view by right-clicking on the project name and selecting **New → Folder**.

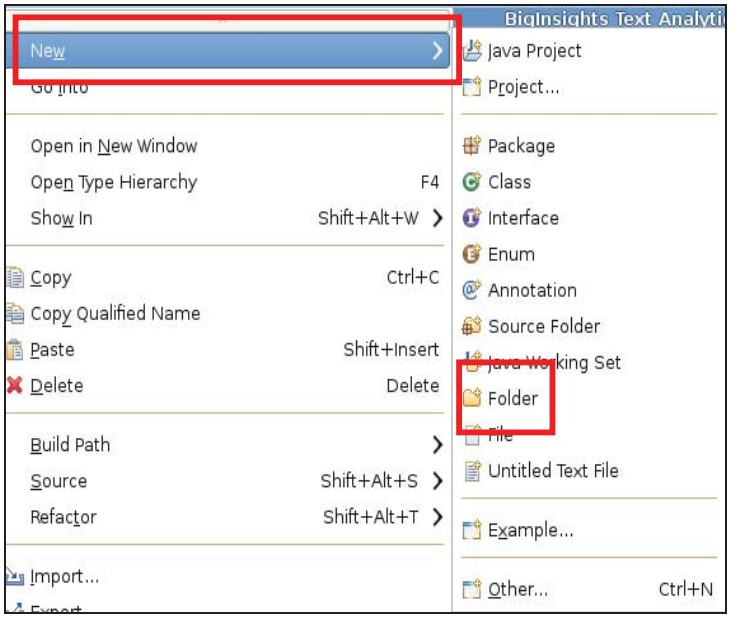

Figure 11 - Create new folder

2. Input **data** as the name of the folder and select **Finish**. We will store the input files here.

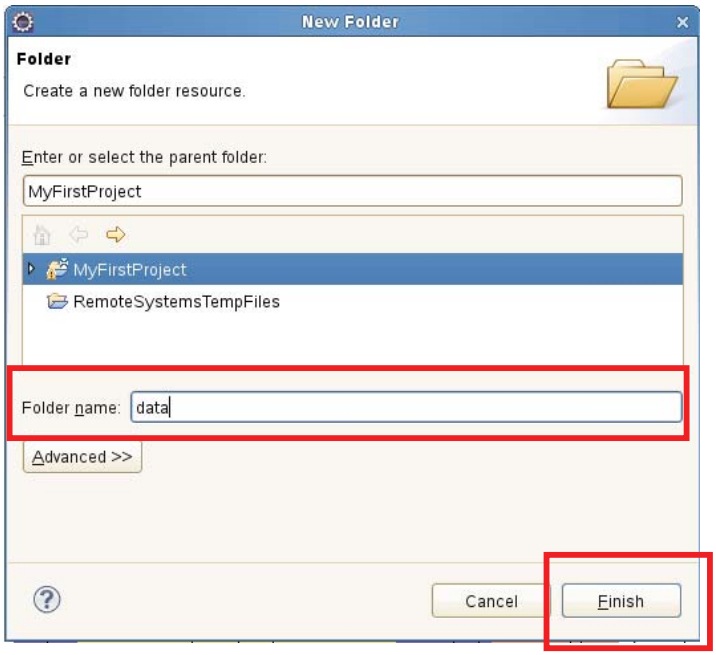

Figure 12 - Input folder name

3. Right-click on the **data** folder, and choose **Import → General → File System**.

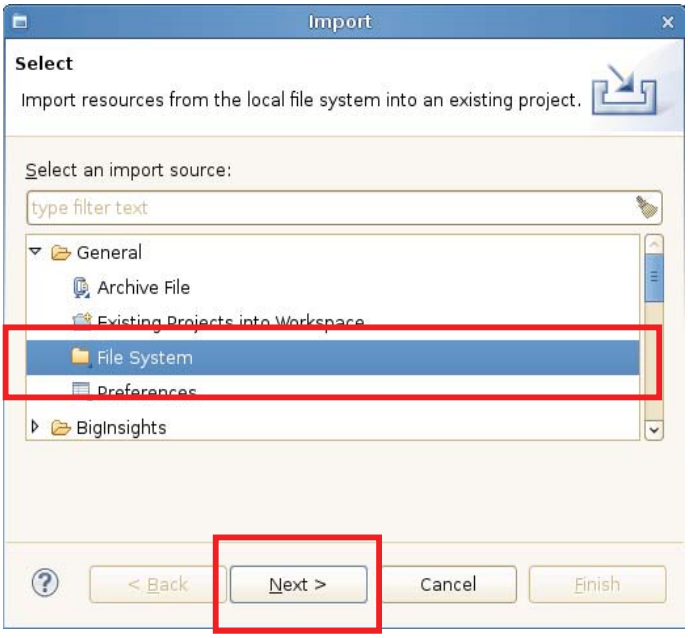

Figure 13 – Select files to import from File System

4. Click **Browse** to navigate to ***/home/biadmin/bootcamp/input/lab04_TextAnalytics*** folder and click **OK**.
5. Back in the Import dialog, expand the **lab04_TextAnalytics** tab in the left pane and select **IBMQuarterlyReports** folder. It will auto-select all text files within the folder. Click **Finish**.

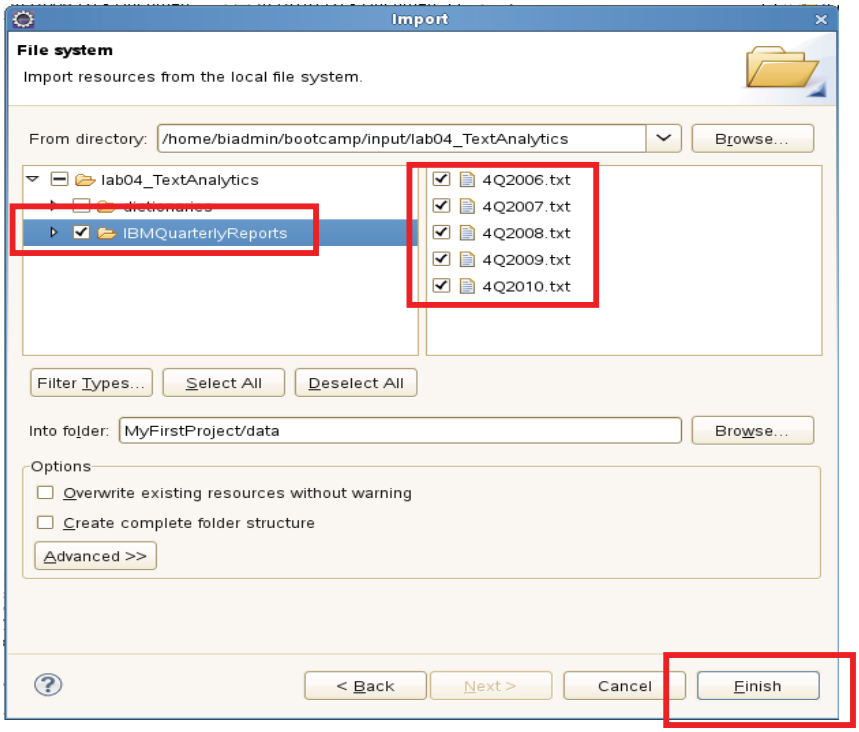

Figure 14 - Importing IBMQuarterlyReports into your project.

After the import, the IBMQuarterlyReports folder containing the actual reports should appear under the *data* folder as shown below.

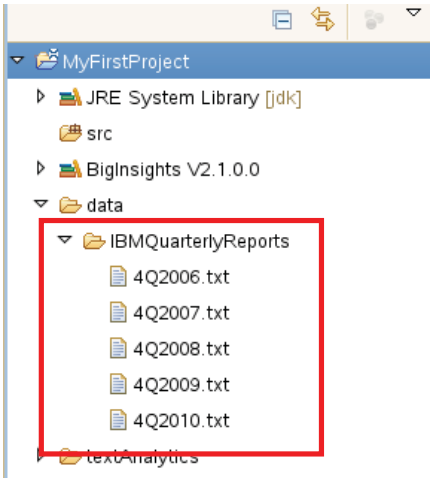

Figure 15 - Input files copied to the data folder.

6. Right-click **MyFirstProject** in **Package Explorer** and select **Properties**.

7. In the *Properties* dialog, ensure that the Text File Encoding is set to UTF-8 by selecting **Resource → Text file encoding → Other → UTF-8.** Click **OK** to close the *Properties* dialog.

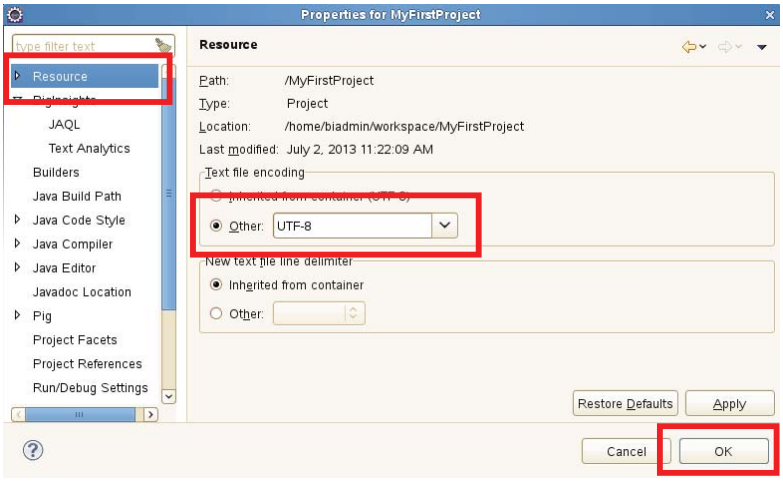

Figure 16 - Select UTF-8 encoding

8. **Repeat step 3** to import files into the **MyFirstProject/textAnalytics/src** folder. Right click the **src** folder in your project, and select **Import**.

   Navigate to ***/home/biadmin/bootcamp/input/lab04_TextAnalytics/*** and check the **dictionaries** folder to select both the folder and its content. Verify that the boxes next to the **dictionaries** folder and **division.dict** file have check marks. Click **Finish**.

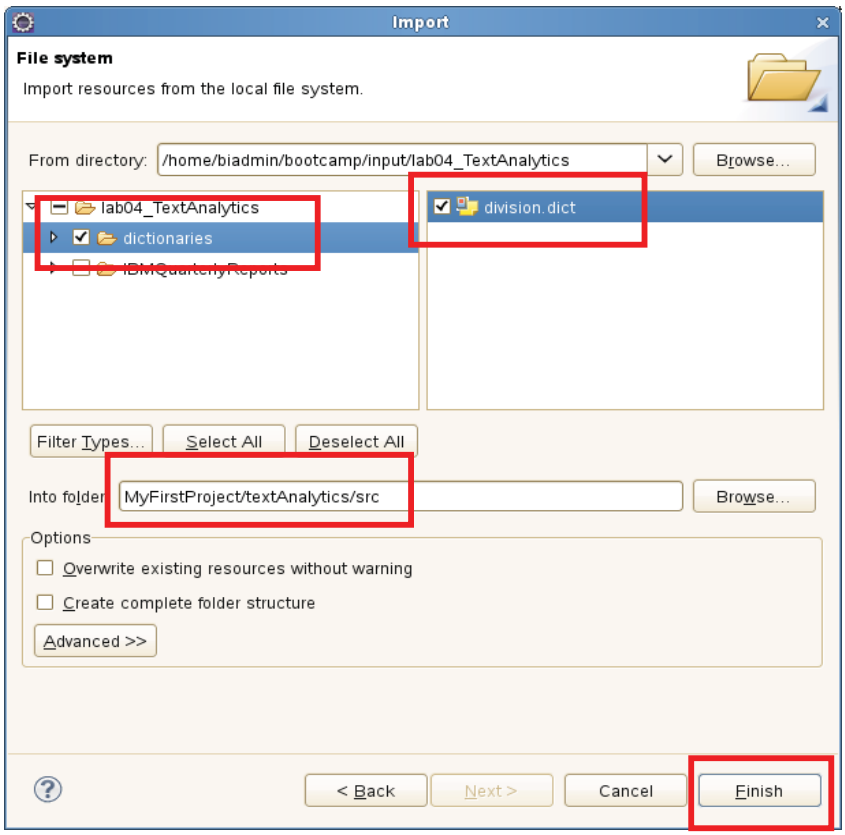

Figure 17 – Dictionaries copied to the src folder

9. Finally, import **solution.aql** into a newly created folder within **MyFirstProject,** name the folder **aql**, this file contains the solution for this lab and might be useful for reference.

   Right click folder **aql**, select **Import**. Browse to ***/home/biadmin/bootcamp/input/lab04_TextAnalytics***, click **OK**, In the Import dialog, select **lab04_TextAnalytics** in the left pane (do not check the checkbox), then check **solution.aql** in the right pane. Verify that the box next to the **lab04_TextAnalytics** folder has a horizontal line, and the box next to the **solution.aql** file has a checkmark. **Click Finish**.

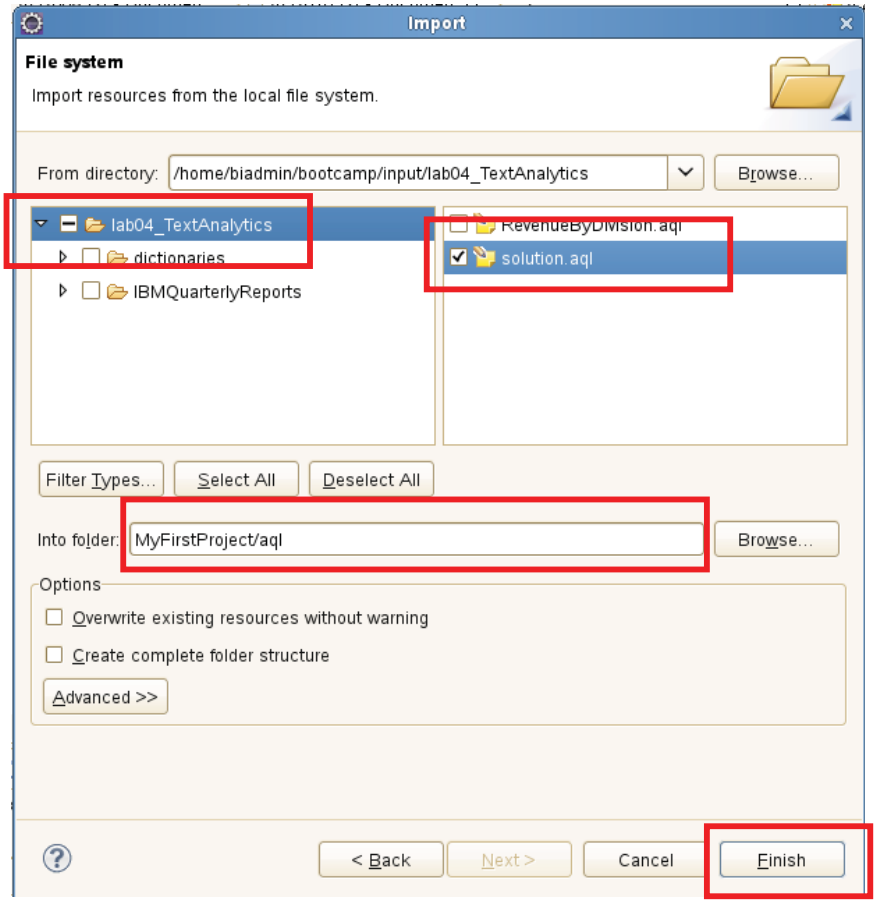

Figure 18 – Importing file solution.aql into your project

10. Your Package Explorer view should look like the following:

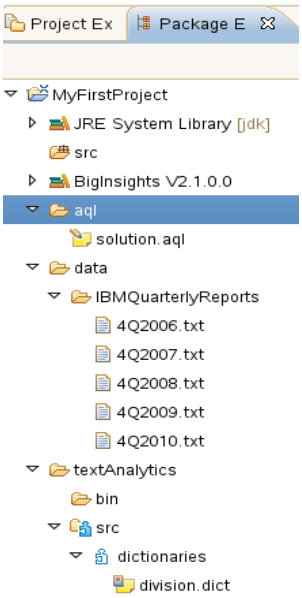

Figure 19 – Project structure after importing required files

Now that we have created a project and imported the documents that require textual analysis, we will begin the text extraction process. The following sections outline the process of creating and running text extractors on a set of documents. We will cover the following steps (these are also outlined within the extraction tasks tab within the BigInsights perspective).

- Step 1: Select Documents
- Step 2: Label Examples and Clues
- Step 3: Develop the Extractor
- Step 4: Test the Extractor

4. Extraction Tasks - Step 1: Select Documents

1. Return to the **Extraction Tasks** view, and select the first step: **Step 1: Select Documents → part a. Select Document Collection**

2. Click **Browse Workspace** and select the folder ***/MyFirstProject/data/IBMQuarterlyReports***. Click **OK**.

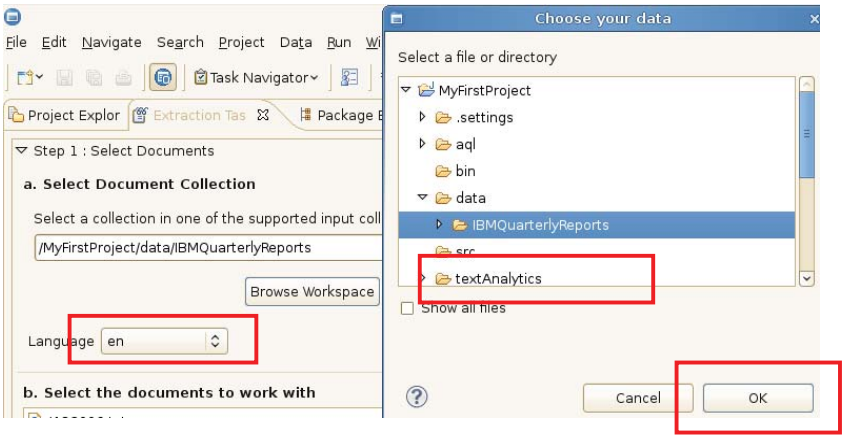

Figure 20 – Selecting input documents to be analyzed.

3. After clicking OK, the path will be added, and the documents in this folder will appear below the window. Select **en** for English from the **Language** drop-down menu.

4. In *Step 1b* of the *Extraction Tasks* view, click on ***/4Q2006*** and click **Open**. The content of the document will open in the editor area.

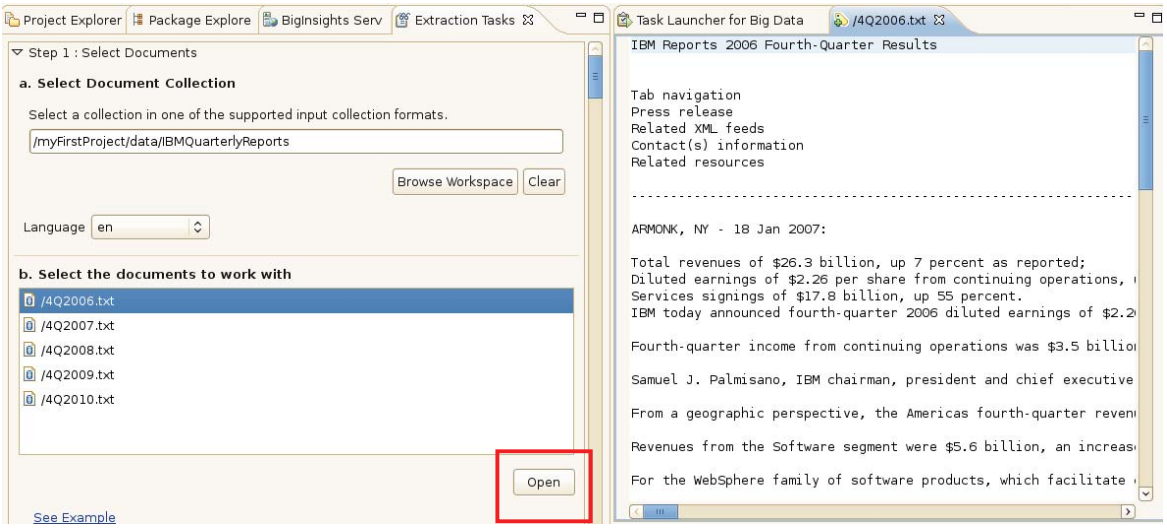

Figure 21 – Selecting documents to be labeled

5. Extraction Tasks - Step 2: Label examples and clues

The objective of this section is to become acquainted with the process of creating an extraction plan by labeling snippets of interest and the clues contained in them. For example "*Revenues from Software were $3.9 billion*" is a **snippet of interest**. Names of division ("*Software*") and revenue numbers ("*3.9*") are **clues within the snippets.**

Observe the automatic creation of an *Extraction Plan* and aql source folders as snippets of interest and clues are identified.

5.1 Part a. Label Snippets of Interest

1. In *step 2 part a. Label snippets of Interest* of the Extraction Task view; our goal is to extract the revenue generated by each business division from each of the quarterly reports. We start by labeling example snippets of revenue by division.

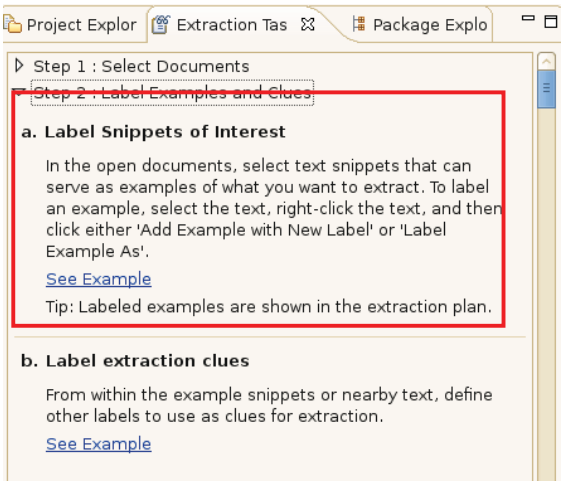

Figure 22 – Labeling documents to identify snippets of interest.

2. In the **/4Q2006.txt** document that is open in the editor, highlight the portion of text that says **"*revenues from Global Technology Services increased 7 percent (…) to $8.6 billion*"**. (You may need to scroll to the right to find this text, or press CTRL+F to open a Search window where you can type the above text to search.) **Right click** the highlighted text, and select **Add Example with New Label**.

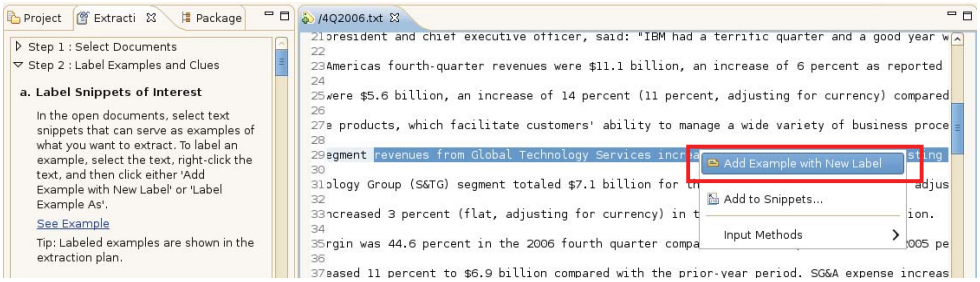

Figure 23 – Selecting text snippets of interest

3. In the **Add New Label** dialog as shown below, type **RevenueByDivision** in the *Label Name* field and click **Finish**.

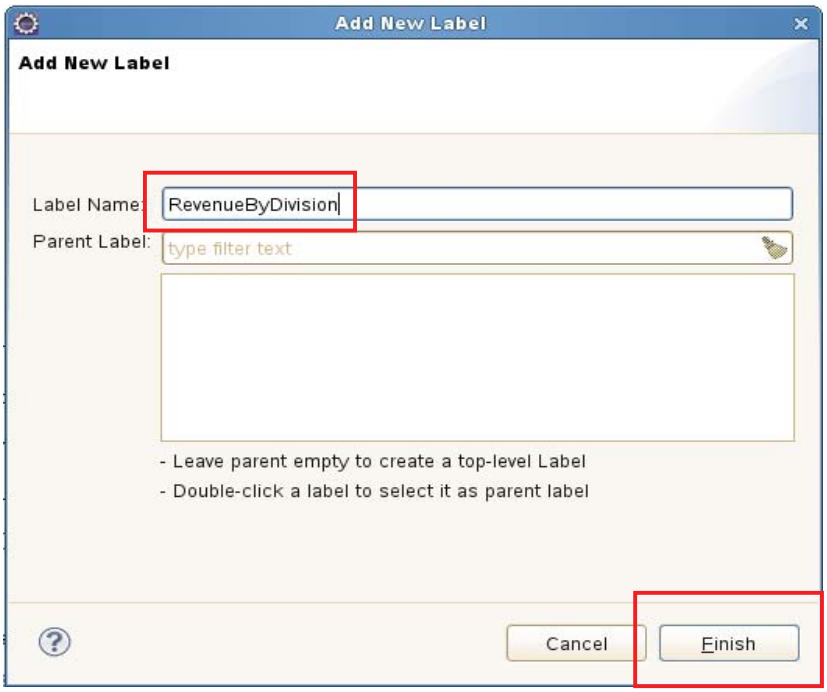

Figure 24 – Add New label

4. The **Extraction Plan** view is populated with the new label and example:

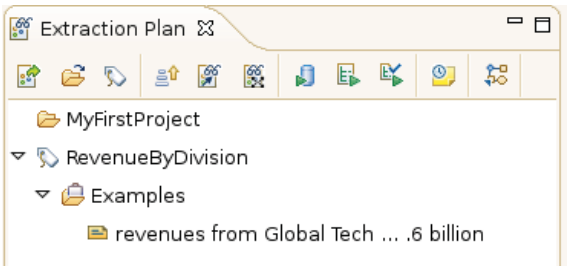

Figure 25 – The extraction plan view in BigInsights text analytics perspective

5. Go back to the **/4Q2006.txt** document in the editor and select one more snippet as an example to be labeled. Highlight the portion of text "***Revenues from the Systems and technology Group (S&TG) segment totaled $7.1 billion***". **Right click** and select **Label Example As → RevenueByDivision**.

① Note the Label Example As option only appears after the first label, RevenueByDivision, (in this example), was created.

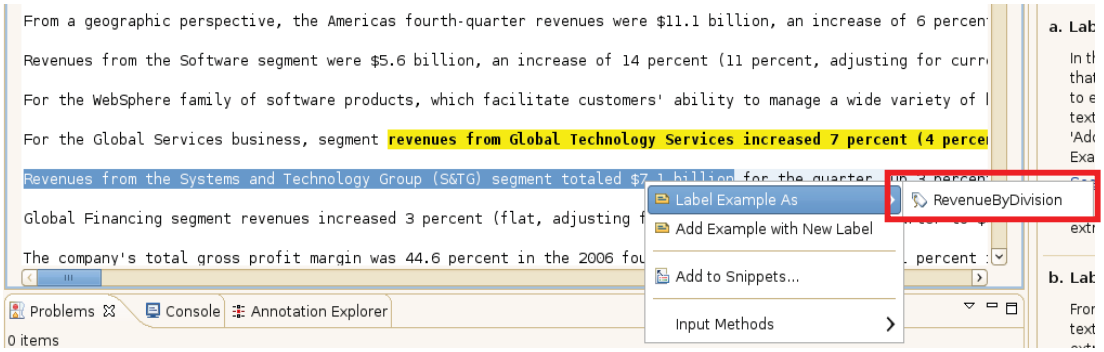

Figure 26 – Adding additional examples to previously created label of a text snippet

6. The new example appears in the ***Extraction Plan*** view.

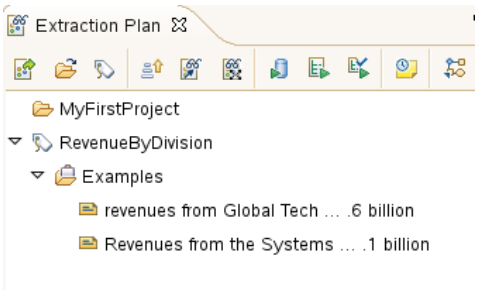

Figure 27 – New label example appears in Extraction Plan

5.2 Part b. Label extraction clues

In this step we label ***clues*** within the snippets we selected as examples. These clues will help extract the information we need.

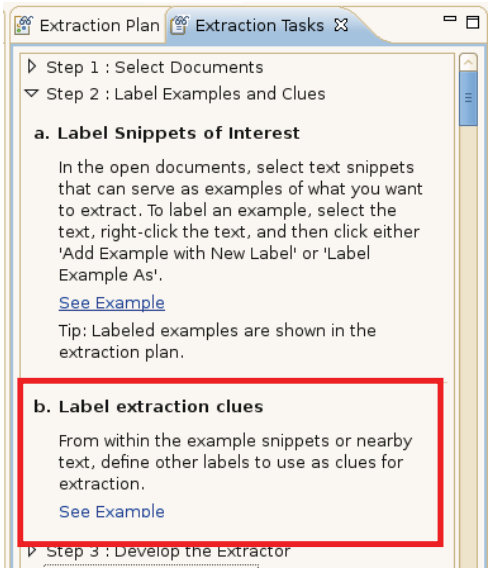

Figure 28 – Labeling clues in snippets of interest

1. Examine the examples labeled as *RevenueByDivision*. In the **Extraction Plan** view, click on the first example under **RevenueByDivision** → **Examples**. You will see the instances highlighted in yellow in the text. **Repeat** this process for the **other example** so you see how the other line is highlighted.

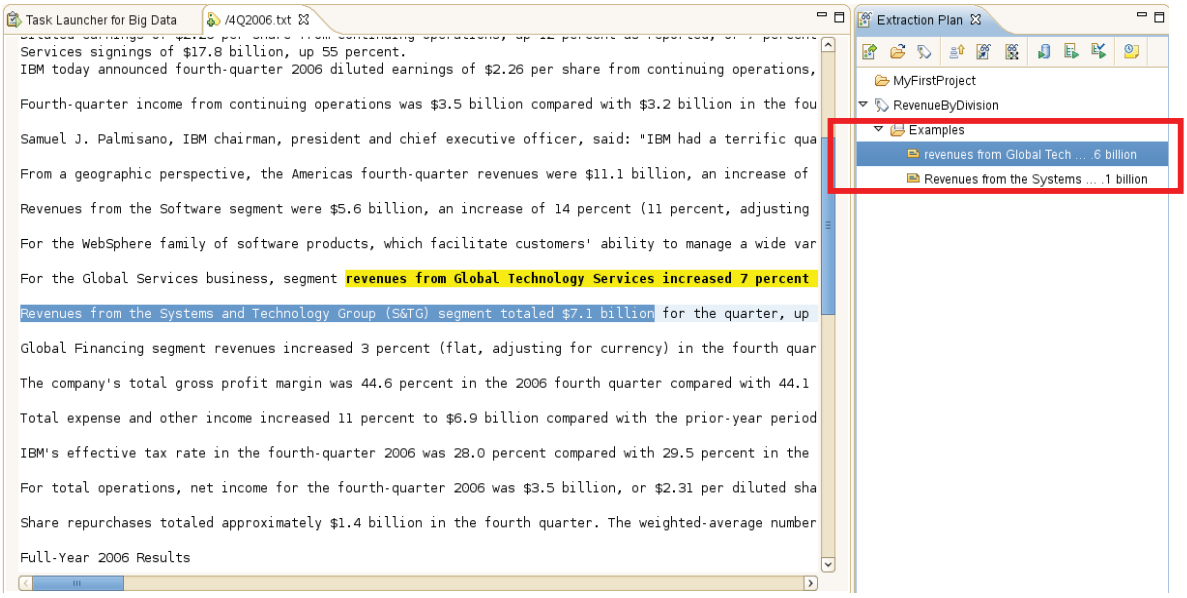

Figure 29 – Reviewing labeled snippets of interest

2. Examine these snippets of text. What types of clues do you think would be useful for extraction?

   revenues from ***Global Technology Services*** increased 7 percent (…) to ***$8.6 billion***

Revenues from the ***Systems and Technology Group (S&TG)*** segment totaled ***$7.1 billion***

Here are a few clues that are useful for this example, and what we are trying to extract:

- The word "**revenues**"
- **Division** names such as Global technology Services, Systems and Technology Group (S&TG)
- **Amounts** such as $8.6 billion, $7.1 billion

We record these clues as examples of **labels** in the *Extraction Plan*. The process of recording is very similar to recording full examples. Let's record the clue for "*revenues*" first.

3. In the **/4Q2006.txt** file, select "***revenues*"** in "*revenues from Global Technology Services …*" **right-click** and select **Add Example with New Label**.

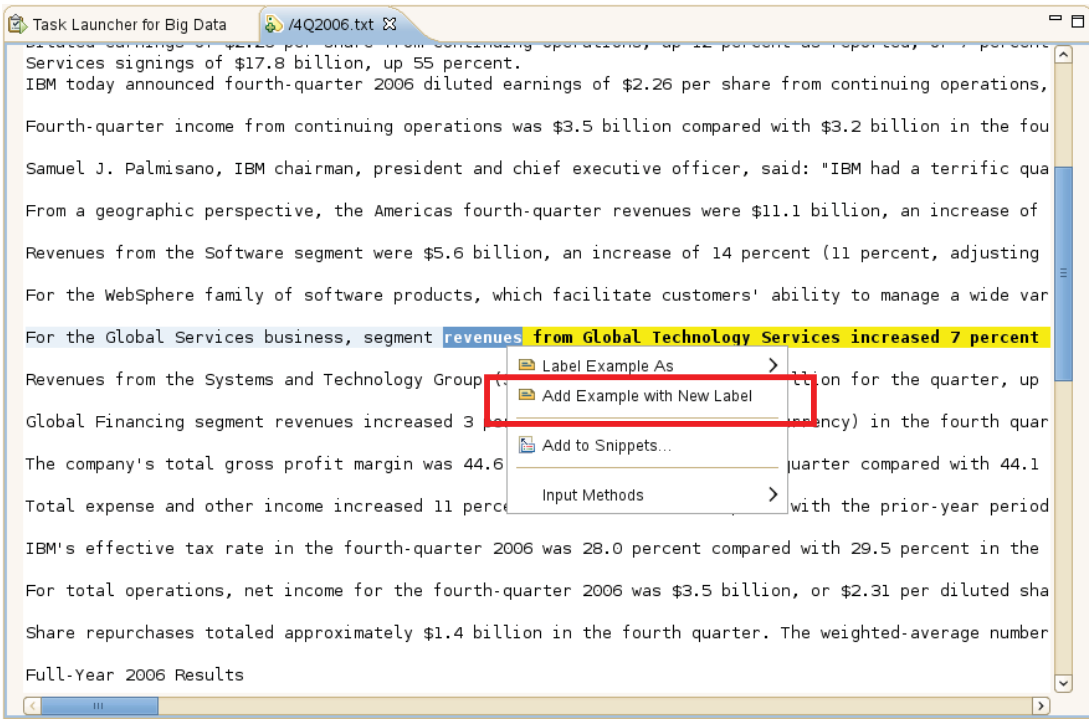

Figure 30 – Identifying clues in the labeled snippets of interest

4. Fill in ***Revenue*** in the *Label Name* field and double-click on ***RevenueByDivision*** to add it to the *Parent Label* field. As shown in the figure below.

Figure 31 – Adding clues to the extraction plan.

5. The new clue is recorded in the *Extraction Plan* view, under *RevenueByDivision* > *Labels*

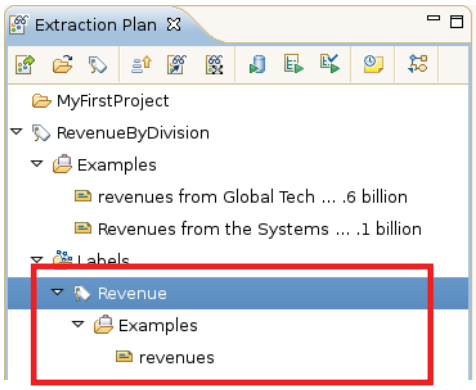

Figure 32 – View of the extraction plan after clue (label) is added

6. Repeat this process to label clues for two other 'Basic Features': *Amount* and *Division*. You would have to create ***two label/clue examples for Amount***, and ***two for Division***:

   i) Highlight ***Global Technology Services*** and right click on it → select **Add Example with New Label** → enter **Division** for Label Name, and **RevenueByDivision** for Parent Label → click **Finish**.

   ii) Highlight ***Systems and Technology Group (ST&G)*** and right click on it → select **Label Example As** → **Division**.

   iii) Highlight ***$8.6 billion*** and right click on it → select **Add Example with New Label** → enter **Amount** for Label Name, and **RevenueByDivision** for Parent Label → click **Finish**.

   iv) Highlight ***$7.1 billion*** and right click on it → select **Label Example As** → **Amount**.

   Upon completing this step, you should have the two labels, *Division* and *Amount*, with two examples each under *RevenueByDivision,* as shown below.

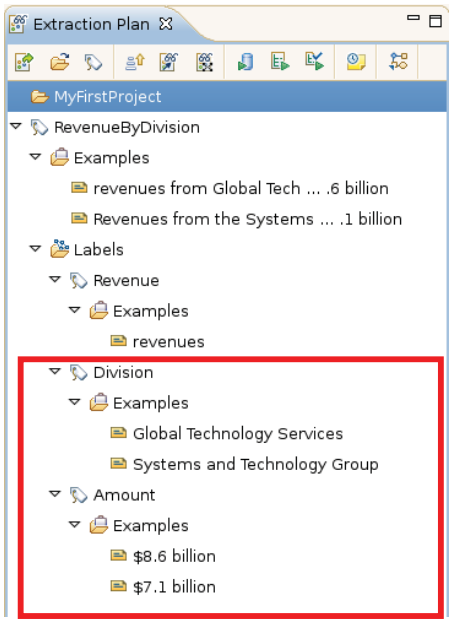

Figure 33 – The Extraction Plan

6. Writing and Testing Extractors for Basic Features

In this section we will write extractors for three basic features: division names, numbers and reference to the term 'revenue'.

- You will get familiar with two tools for extracting the basic features:
  - Use of regular expressions in the aql Extract statement. We will locate the numeric value of the revenue of a division, and occurrence of the clue 'Revenue' in a statement using regular expressions.
  - Use of dictionaries in the Extract statement, both inline dictionaries and dictionary files. We will use a dictionary to identify names of the IBM divisions

6.1 Extraction Tasks - Step 3: Develop the Extractor

In Step 3 of the *Extraction Task* view, we will create the extractor for 'division' using the dictionary ***division.dict***, which we copied into the AQL folder during setup.

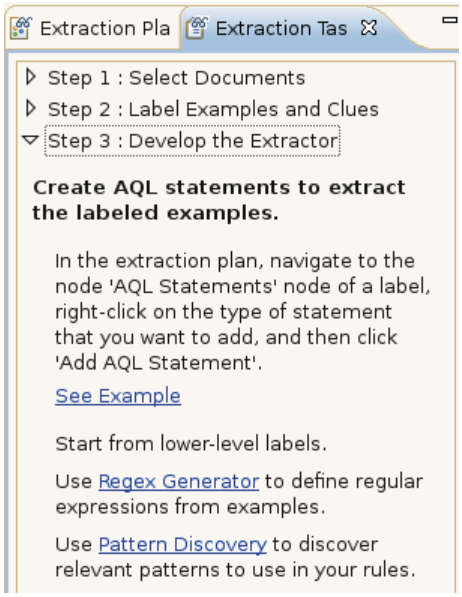

Figure 34 – Third step of the task flow, writing the extractors in the extraction plan

1. Add a rule to extract division names. In the **Extraction Plan** view, navigate to ***RevenueByDivision*** → ***Labels*** → ***Division.*** Right-click on the **Division** label and select **New AQL Statement**, and choose ***Basic Feature AQL Statement***.

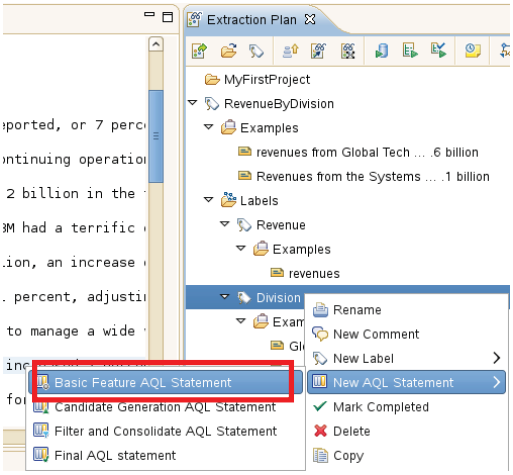

Figure 35 – Selecting the extractor to be written for basic features of the labeled clues in the extraction plan

2. In the *Create AQL Statement* dialog, use ***Division*** as the view name, and then for the AQL statement type, choose ***Dictionary*** since we have the dictionary *division.dict* that we loaded earlier in the lab and specify the aql script name as Division_basic (this names the aql file containing the extractor code). Select Output view, to be able to see results. Click **OK**.

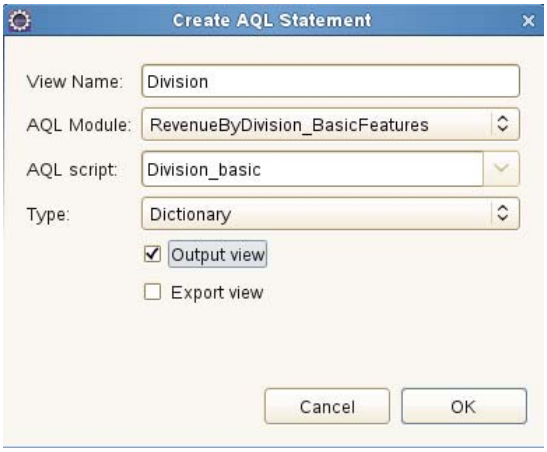

Figure 36 – Specifying the extractor name and extractor type

The file *RevenueByDivision/Division_basic.aql* opens in the editor and a template for the ***extract dictionary*** *AQL* statement is automatically added (do not worry about the parse error, you will fix that shortly):

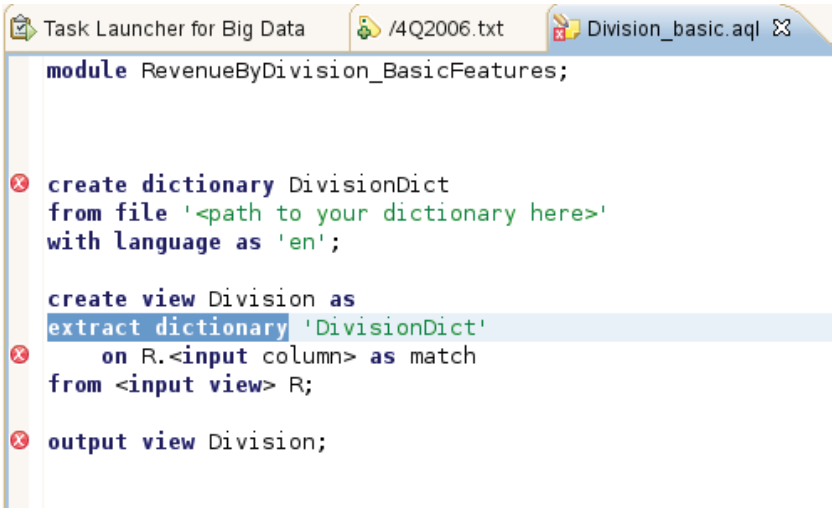

Figure 37 – Template for the extract statement for dictionary of division names

A few things to notice about the template:

- It contains a CREATE DICTIONARY statement defining a dictionary DivisionDict from an external file – you will need to fill in the template for *<path to your dictionary here>* with the path to the dictionary file in your project.
- The language used for matching the dictionary is 'en' (for English) – (the same language was specified in Step 1.a of the Extraction Tasks view)
- It contains a template for an ***extract dictionary*** statement for matching the previously defined *DivisionDict* dictionary. You will need to fill in the templates for *<input column>* and *<input view>*. When dealing with text documents from the file system we use the input view keyword ***'Document'*** to reference all input documents and the input column keyword ***'text'*** to reference the text of the input documents. ***Document.text*** is the correct way to refer to the content of any input document.

The corrected template should therefore look like:

```
module RevenueByDivision_BasicFeatures;

create dictionary DivisionDict
from file '../dictionaries/division.dict'
with language as 'en';

create view Division as
extract dictionary 'DivisionDict'
      on R.text as match
from Document R;

output view Division;
```

Note: The file path reference stated above has to match the path to division.dict and is relative to the root of the module in which the create dictionary statement is issued.

Dictionaries are usually stored in files to keep them separate from the aql code. The dictionary for 'Division' is shown below.

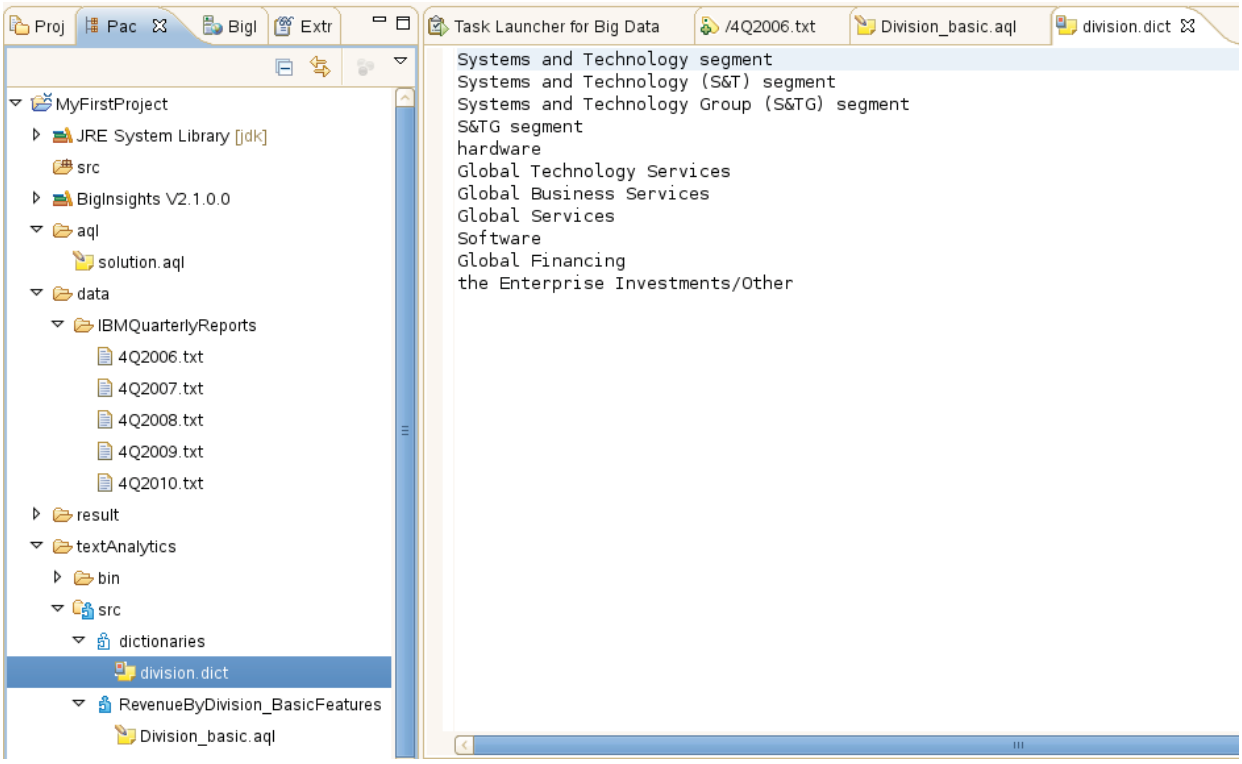

Figure 38 – Division names dictionary

3. Save the aql files by selecting **File** → **Save** from the top menu bar in Eclipse, or press or ctrl+s on your keyboard.

6.2 Extraction Tasks - Step 4: Test the Extractor

The results of the extraction are displayed in the *Annotation Explorer* as shown below.

1. Now that we have created an extraction rule for one basic feature – 'Division', it is time to test it.

Go to *Step 4: Test the Extractor* in the *Extraction Task* view. Choose the last option, ***Run the extraction plan on the set of documents that are labeled***, as shown in the figure below. **Click** on the ***icon*** beside it.

The results of the extraction are displayed in the ***Annotation Explorer*** view as shown below.

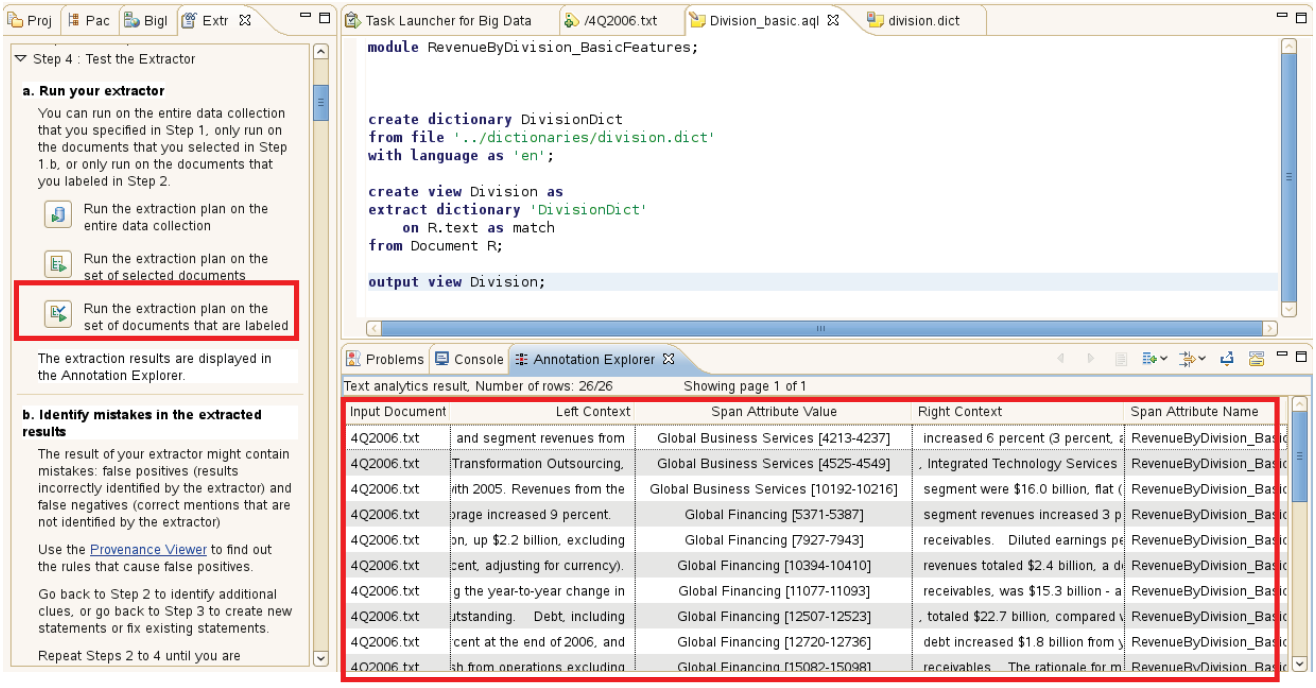

Figure 39 – *Fourth step of the task flow, testing the extractor and the Annotation Explorer view*

2. The Span Attribute column in the middle of Annotation Explorer shows the *division* names picked up by the extractor. You will notice towards the bottom in the explorer view that "*software*" is being picked up incorrectly as a division name. This can be fixed by modifying the Extract Dictionary statement to use the '***Exact'*** flag to ensure that the text string matches the dictionary entry exactly, including case.

   Modify the create view statement for Division_basic.aql as shown below, ***save the new aql*** and ***run the extraction plan again*** in the same way as before.

```
create view Division as
extract dictionary 'DivisionDict'
  with flags 'Exact'
  on R.text as match
from Document R;
```

3. ***Double-click*** on one of the rows for ***Systems and Technology Group*** in ***Annotation Explorer*** to see the position of the extracted text in the document. It is highlighted in blue in the edit area. An annotation tree view pops up on the

right side. Check the *division annotator box*, and all division entities extracted from the document will be highlighted in the document as shown in the figure below.

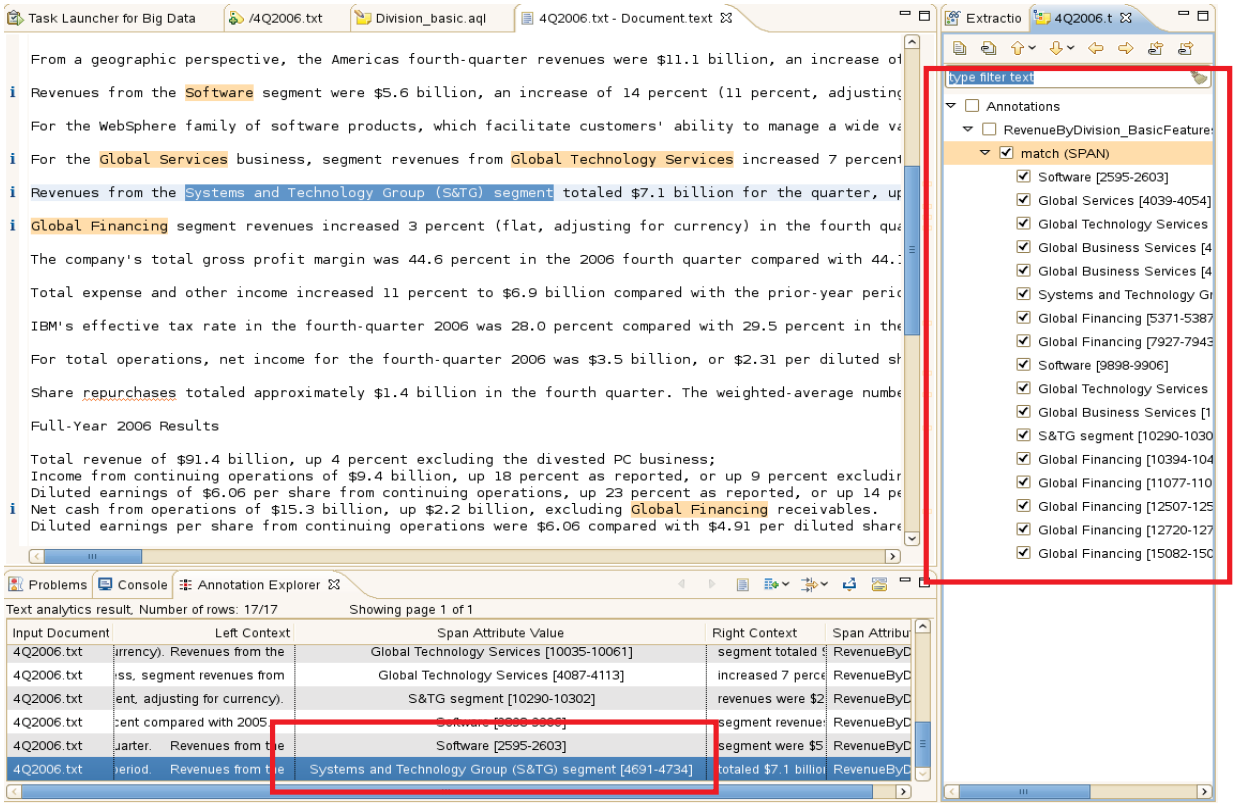

Figure 40 – Visual examination of the extracted basic features (Division names)

4. Back in the **Extraction Tasks** view, click the icon beside step 4.a "***Run the extraction plan on the entire data collection***". This will display the results extracted from *all* the documents in your collection (Step 1.a of the Extraction Tasks), not just the document 4Q2006.txt that you have labeled.

The button "***Run the extraction plan on the entire data collection***" of Step 4.a of Extraction Tasks performs the same task as if you would configure a Text Analytics run configuration manually (in Package Explorer, right click on project and select Run Configurations. In the wizard, create a new Text Analytics configuration and specify the input data collection as the collection selected in Step 1.a of Extraction Tasks view.)

6.3 Writing the basic extractors for Number and Revenue

We will now create extractors for two more *basic features* – Number and Revenue. We are looking for numbers - 12, 12.5, 27.2. Later we will use this basic feature in a *pattern* to identify instances of an amount with a unit - $1.2 billion or $12.5 million. This is a powerful feature of AQL that lets us identify basic features based on a regular expression or a dictionary and then use a pattern to put the basic features into context.

Note: To refer to the complete code for these extractors, you can refer to the solution provided in the *solution.aql* file within the *aql* folder.

I. Regular expression rule to identify 'Number'

1. In the **Extraction Plan** view, navigate to ***RevenueByDivision*** → ***Labels*** → ***Amount***. Right-click the label **Amount** and select ***New AQL Statement***. *Select* ***Basic Features AQL statement.***

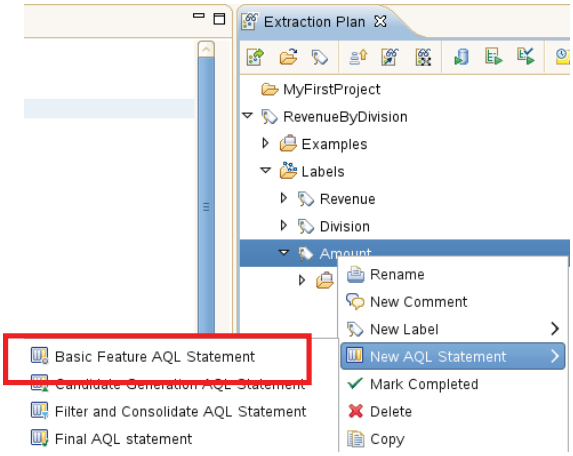

Figure 41 – *Create new AQL statement for an extractor for Amount*

2. In the *Create AQL Statement* dialog enter "***Number***" as the view name and select ***Regular Expression*** as the *type*. As we plan to use this view later in a pattern, check the box '***Export view'***. The *export view* statement will be generated in the template for the aql script. This makes the view available outside the RevenueByDivision_BasicFeatures AQL module. Name the AQL script ***Number_basic***. After you have entered all that information click the **OK** button.

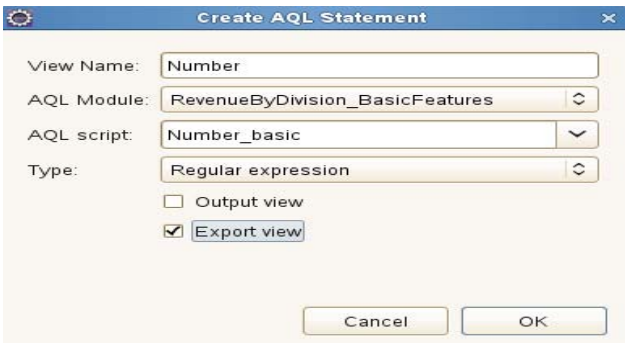

Figure 42 – *Create new AQL statement for an extractor for Amount*

You will get the following template in the ***Number_basic*** file within the aql editor pane. It will show certain errors.

```
create view Number as
extract regex /<regular expression specification>/
          on R.<input column> as match
from <input view> R;

export view Number;
```

3. The following errors need to be fixed:

    1) the regular expression to look for a contiguous sequence of digits followed by an optional decimal point together with another sequence of digits,

    2) the <input column> for input files is text (as we saw before), and

    3) <input view> name for input files is always Document.

    4) Save the aql files by selecting **File** → **Save** from the top menu bar in Eclipse , or press or ctrl+s on your keyboard

```
module RevenueByDivision_BasicFeatures;

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


export view Number;
```

Regular expression `/\b\d+(\,\d{3})*(\.\d+)?\b/` is more robust to use for numbers than the simple one `/d+(\.\d+)?/` given above, though the simple regular expression suffices for this exercise.

II. Regular expression rule to identify Revenue:

Now we create another regular expression based extract statement called "*Revenue*" to identify Revenue clues. Basically, the mention of the term revenue alone or the mention of the term revenue along with mention of first/second/third or fourth quarter, is a clue that we may be talking about the revenue of a division if the division also happens to be mentioned in close proximity of the revenue clue.

1. Navigate to ***RevenueByDivision*** → ***Labels*** → ***Revenue.*** Right-click the label **Revenue** and select ***New AQL Statement***. *Select* ***Basic Features AQL statement.***
2. Follow the steps stated in the previous section and fill in the pop-up box for 'Create AQL Statement'.
    - Enter "***Revenue***" as the *view name* and choose ***Regular Expression*** as the *type*.
    - Enter the aql script name as ***Revenue_basic*** and check the box '***Output view***'. Click the **OK** button.

```
create view Revenue as
extract regex /<regular expression specification>/
      on R.<input column> as match
from <input view> R;

output view Revenue;
```

4. The <input view> and <input column> have to be filled in as we did before for the *number* view. An implementation for the *regex* is suggested below. The final extractor would look as follows.

```
module RevenueByDivision_BasicFeatures;

create view Revenue as
extract regex /((.\s*)?(first|second|third|fourth)\s*-\s*quarter)?\s*(revenues?)/
      with flags 'CASE_INSENSITIVE'
      on between 1 and 5 tokens in R.text
      return
            group 0 as match
from Document R;
output view Revenue;
```

5. At this point, the *Extraction Plan* view should look as follows:

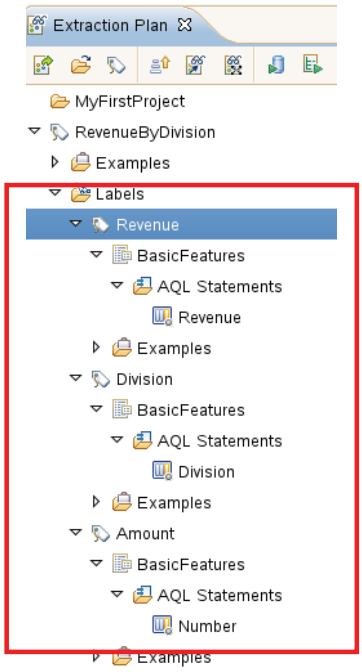

Figure 43 – Extraction plan view after all extractors for basic features have been written.

6. To see the extractions of the three basic features (Division, Number and Revenue), ensure that all aql files have been saved. Navigate to ***File*** → ***Save*** *(or ctrl+s)* for each file.

   Then navigate to the **Extraction Tasks** view, as shown in section 6.2. Select ***Step 4: Test the Extractor*** and choose the last option, ***Run the extraction plan on the set of documents that are labeled*** and click the ***icon*** beside it.

7. The results of the extraction are displayed in the *Annotation Explorer* view. Double clicking on one of the rows in the annotation explorer will open the labeled document in the edit window. The extracted basic features can be highlighted by checking the corresponding boxes in the annotations window on the right.

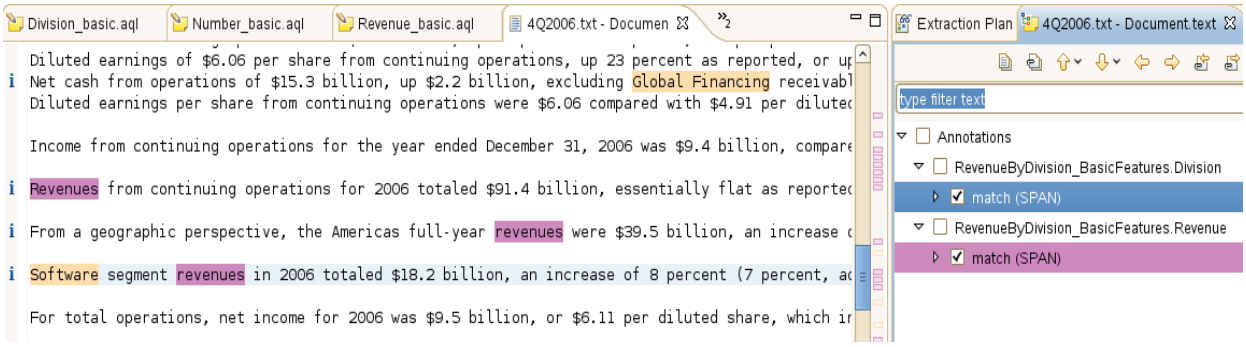

Figure 44 – *Visualization of all the basic features extracted*

Note: Annotation matches for "Number" view are not displayed because we asked to export the view and not output.

The aql code for the extractors for the three basic features can be found under the folder **RevenueByDivision_BasicFeatures** in Package Explorer. The code for all the files is listed below:

```
create dictionary DivisionDict
from file '../dictionaries/division.dict'
with language as 'en';

create view Division as
extract dictionary 'DivisionDict'
   with flags 'Exact'
   on R.text as match
from Document R;

output view Division;

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

export view Number;

create view Revenue as
extract
regex /((.\s*)?(first|second|third|fourth)\s*-\s*quarter)?\s*(revenues?)/
      with flags 'CASE_INSENSITIVE'
      on between 1 and 5 tokens in R.text
      return
            group 0 as match
from Document R;

output view Revenue;
```

7. Writing Extractors for Concepts based on Basic Features

In this section we will extract two ***concepts***:

1. Mention of revenue of an IBM division and
2. The reported revenue of that division.

We will make use of two aql constructs; the Extract Pattern statement, and the Select statement. The three basic features outlined in the previous section will be used as inputs.

- The reported revenue is a simple pattern – a '$' symbol followed by a number, followed by a revenue unit. We will use the extract pattern statement together with the view ***Number*** we previously created and a list of revenue units.
- The mention of revenue of a division has two different patterns: 1) mention of division name followed by some variant of the word 'revenue', 2) or the word 'revenue(s)' followed by mention of the division name. We will identify both patterns and use the 'union all' construct of aql to combine them into a single view.

7.1 Create Extractor for AmountwithUnit

We will add a *candidate generation rule* for *Amount*. This uses *Sequence Pattern* from the *Extract* statement.

1. In the **Extraction Plan** view, navigate to ***RevenueByDivision →Labels → Amount***, and right-click on the **Amount** Label. Select ***New AQL statement*** and the option ***Candidate Generation AQL Statement***.

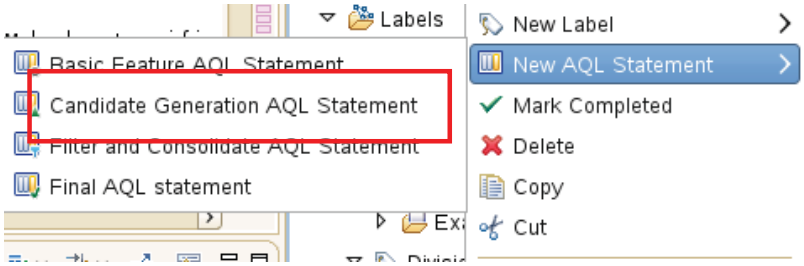

Figure 45 – New AQL Statement to create extractor for candidate generation

2. Fill in the dialog as shown in the following figure, with view name **"AmountwithUnit**", AQL script "**AmountwithUnit_concept**", and for Type select "**Pattern**". Also, remember to check "**Output view**". Then click the **OK** button.

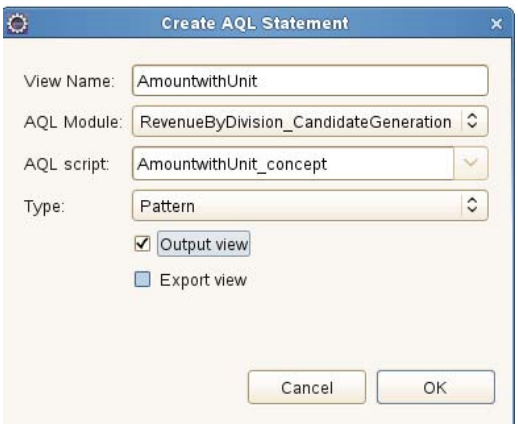

Figure 46 – Assigning name and extractor type to extractor for candidate generation

3. The new view appears in the Extraction Plan, and the following template is added to ***AmountwithUnit_concept.aql*** in the *aql editor pane.*

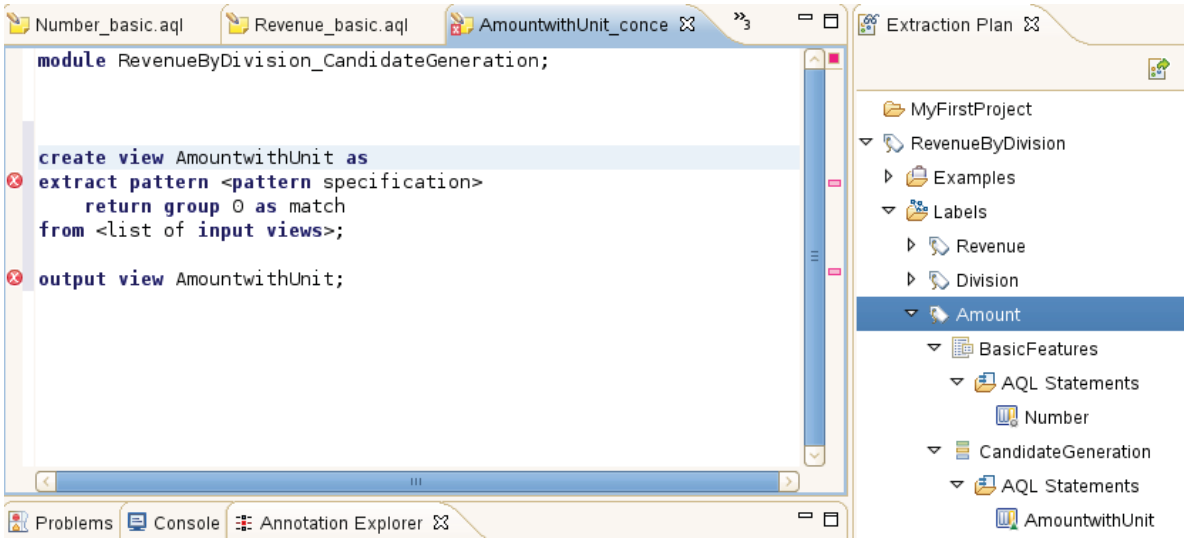

Figure 47 – Template for extract pattern extractor

> ⓘ **Note: Label names are case-sensitive, ensure that names are used correctly in later sections.**

4. Here we will use the *basic feature* that we previously created: ***Number***.

5. Add the following lines of code under the module definition for **RevenueByDivision_CandidateGeneration**, to import the required view into the current module.

```
module RevenueByDivision_CandidateGeneration;
--<add the following line>
import view Number from module RevenueByDivision_BasicFeatures as Num;
```

> ⓘ **Note: An import statement puts objects in the context of the current module. The import statement is only used to import objects from other modules, not from the current module. Remember that an object that is declared in an AQL file of the module is visible to any other AQL file in that same module.**

6. After the import, we fix the statement so it extracts the pattern we want. Your final aql file should look similar to the one below. **Save** the newly modified file.

```
module RevenueByDivision_CandidateGeneration;

import view Number from module RevenueByDivision_BasicFeatures as Num;

create view AmountwithUnit as
extract pattern '$'<N.match> ('million'|'billion'|'trillion')
      return group 0 as match
from Num N, Document R;

output view AmountwithUnit;
```

7. Run the extractor by navigating to the Extraction Tasks view and selecting ***Step 4.a*** in the ***Extraction Tasks*** *view* (3rd option “Run the extraction plan on the set of documents that are labeled”), will produce the following AmountwithUnit results. ***Double-click*** on a row in Annotation Explorer to see the panel on the right, and check the box for ***AmountwithUnit.***

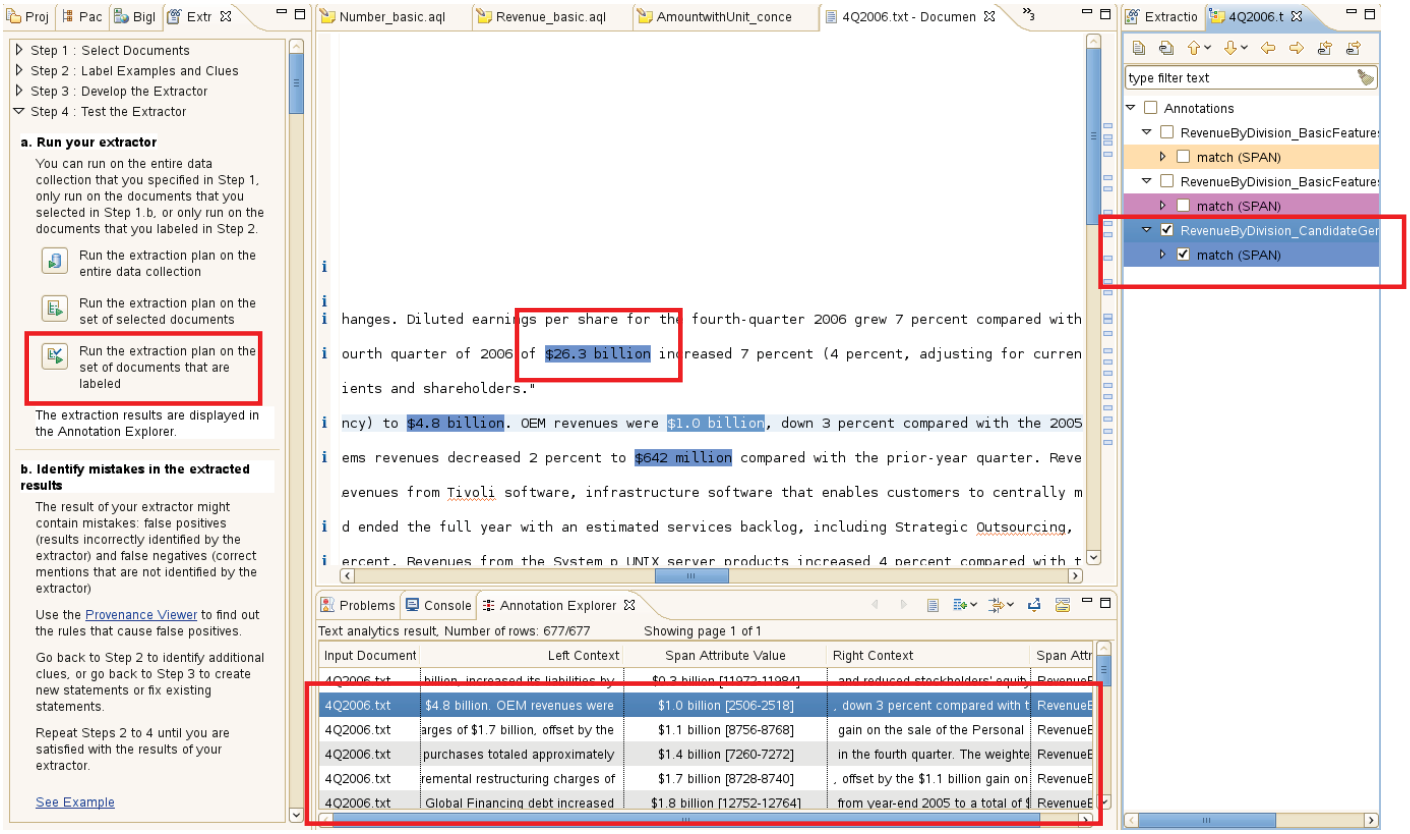

Figure 48 – Results for AmountwithUnit extractor, sequential occurrence of a number followed by a unit

7.2 Create Extractor for RevenueByDivision

At this point we need to add a Candidate Generations rule for RevenueByDivision to identify occurrence of a division name followed by a reference to revenue, with at most one token in between. For example “**Global Financing segment revenues** increased 3 percent” or ”**Software segment revenues** in 2006 totaled $18.2 billion”
For this section, the AQL script was already created. You will need to import the aql file.

1. From **Package Explorer** view, under MyFirstProject, navigate to **textAnalytics** → **src** and right click **RevenueByDivision_CandidateGeneration**.
2. Select **Import** → **File System** and **Next**. Browse to **/home/biadmin/bootcamp/input/lab04_TextAnalytics**
3. Select **RevenueByDivision.aql.** Verify that the box next to **lab04_TextAnalytics** folder has a horizontal line and the box next to **RevenueByDivision** file has a **checkbox**. Click **Finish.**

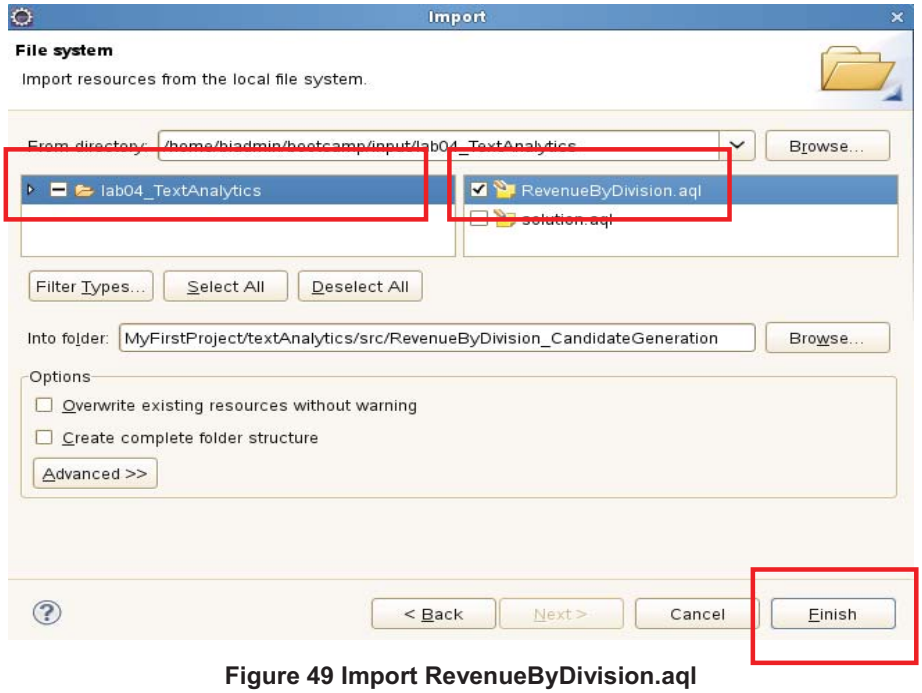

Figure 49 Import RevenueByDivision.aql

4. In order to use the two views from other AQL module, they have to be available. Change the definition for ***Revenue*** and ***Division*** to *export* the view rather than use as output. Edit **Division_basic.aql** and change "`output view Division`" statement to "`export view Division`". Same for **Revenue_basic.aql.**

5. Before running the extraction plan, **save all the scripts**.

6. In the **Extraction Tasks** view, select ***Step 4: Test the Extractor***, choose the last option, ***Run the extraction plan on the set of documents that are labeled*** and click the ***icon*** beside it. You will see mentions of division names reference with revenue, and their revenues as shown below.

ⓘ Note: Select the match spans for Division and Revenue in the Right Pane to highlight the extracted text

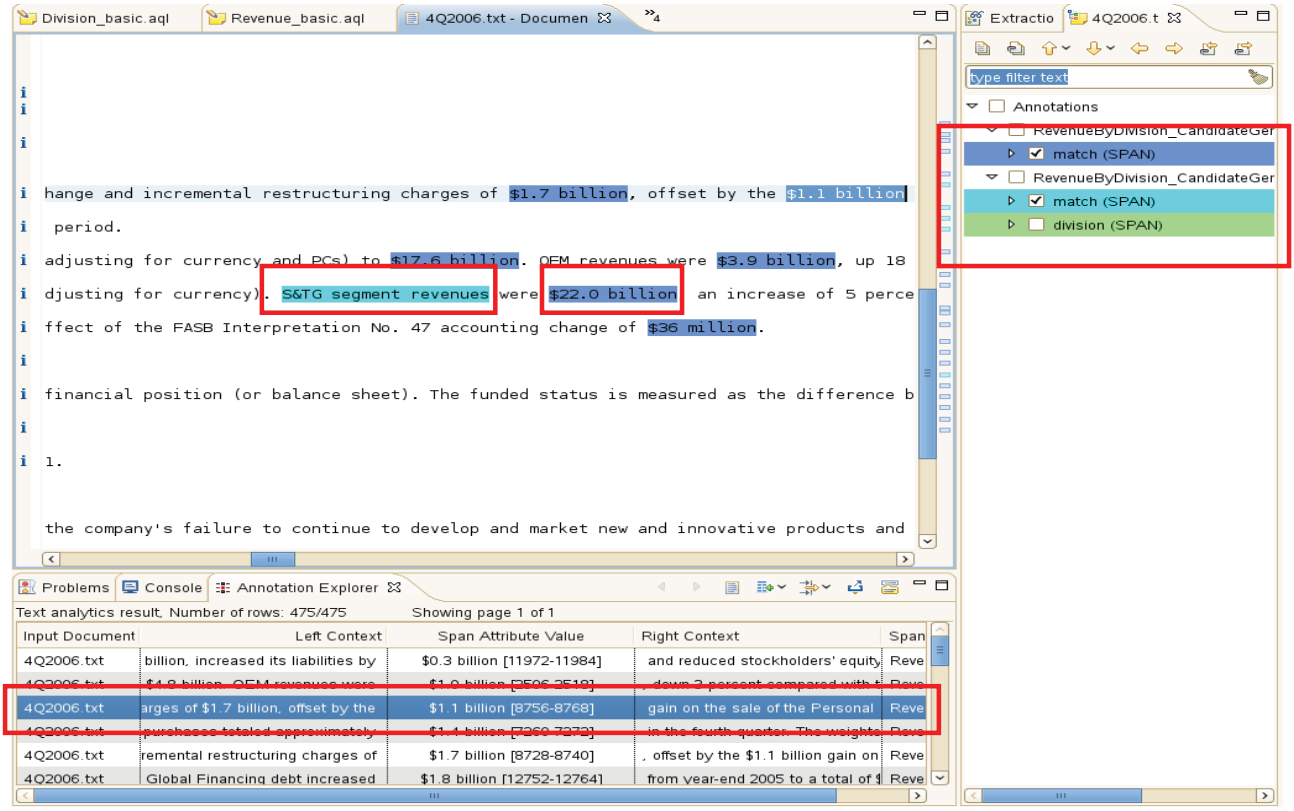

Figure 50 – Visualizing the RevenueByDivision and AmountwithUnit entities extracted

The two red rectangles in the figure highlight the missing association between the division name and its revenue, which we will establish in the next exercise. Key to this association is that the revenue amount follows the reference to revenue of a division within a few tokens.

7.3 Extending the extractor for RevenueByDivision by including AmountwithUnit information

In this section we will create one view for the reference to the revenue of a division and the reported revenue of that division. This will be done by using the '*select'* statement of AQL to combine the RevenueDivision and AmountwithUnit views. The pattern we will look for is revenue of Division followed by AmountwithUnit with a maximum of 15 intervening tokens.

1. Add a Candidate Generation rule for **DivRevenueWithAmount** by navigating to ***Extraction Plan*** and right-clicking on ***RevenueByDivision.*** Select ***New AQL Statement → Candidate Generation AQL statement***. Fill in the fields as follows: the view name **"DivRevenueWithAmount**", AQL script "**DivRevenueWithAmount**", and type "**Select**". Also, remember to check "**Output view**". Then click the **OK** button:

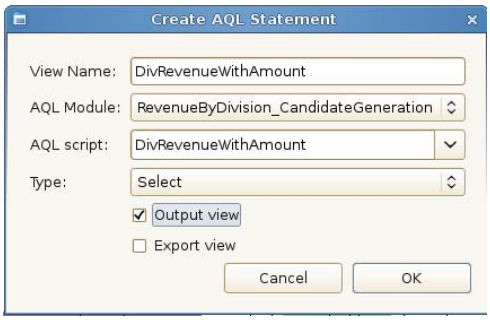

Figure 51 – Creating view DivRevenueWithAmount

2. List the two views needed as the input in the 'from' clause, `RevenueByDivision and AmountwithUnit.` In the 'where' clause, we specify that a maximum of 15 intervening tokens are allowed between the reference to a division and the revenue number. Finally in the select clause we select not only the spans of interest, but also the values using the GetText() function. The final code for this view is:

```
module RevenueByDivision_CandidateGeneration;

create view DivRevenueWithAmount as
select CombineSpans(R.match, A.match)
     as match,
     R.division as division,
     A.match as revenue,
     GetText (R.division) as divName,
     GetText (A.match) as revText
from RevenueByDivision R, AmountwithUnit A
where FollowsTok(R.match, A.match, 0,15);

output view DivRevenueWithAmount;
```

3. Save the file and in **Extraction Tasks**, select ***Step 4: Test the Extractor***, choose the last option, ***Run the extraction plan on the set of documents that are labeled*** and click the ***icon*** beside it.

   Double Click one of the entries inside the Annotation Explorer.

   You will see the ***DivRevenueWithAmount*** view on the panel on the right hand side. Within that panel, toggle the checkboxes for *match, division*, and *revenue* fields to see the corresponding extracted entities as shown in the figure below.

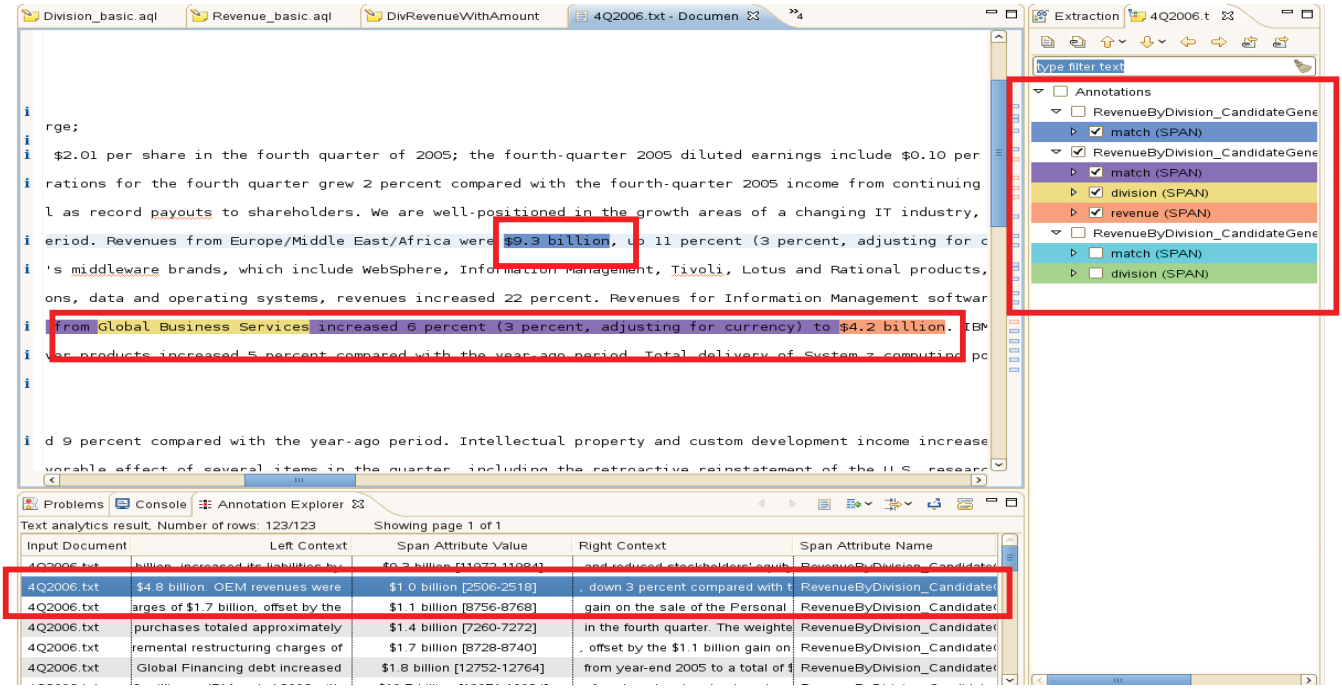

Figure 52 – *Division and* revenue *extracted as a single concept*

8. Analyzing Extracted Results with Annotation Explorer

The Annotation Explorer has a row entry for every span identified in every output view. In this section we will first display one of the output view tables defined in the previous sections. Then we will use filters to display a selected subset of these rows.

8.1 Displaying Output View Table

1. To display an output view table, click the down arrow in the highlighted area in Figure 53 below, and select ***'RevenueByDivision_CandidateGeneration.DivRevenueWithAmount'*** view (in case if you do not find the icon, on the top menu bar, select ***Window → Reset Perspective*** first). This will generate the output view table as shown in figure 54. Each row in Figure 53 is a relation, or tuple, of the output view. The columns of the table are the fields in the tuple. For fields of type span, the text corresponding to SPAN is also included.

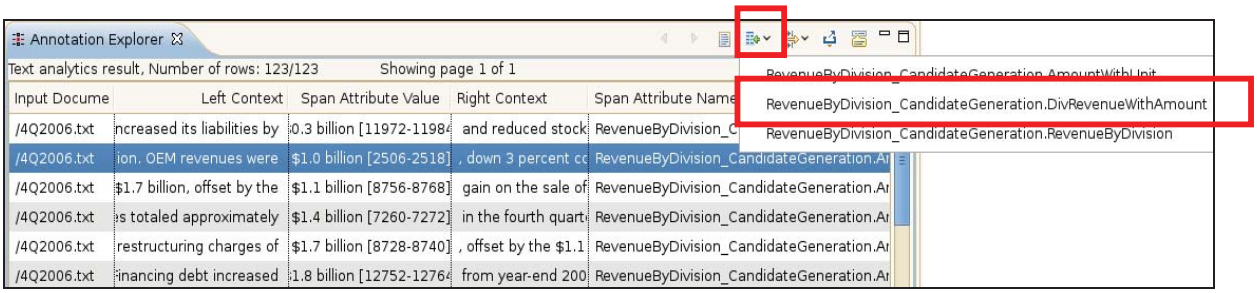

Figure 53 – Setting up display of an Output View

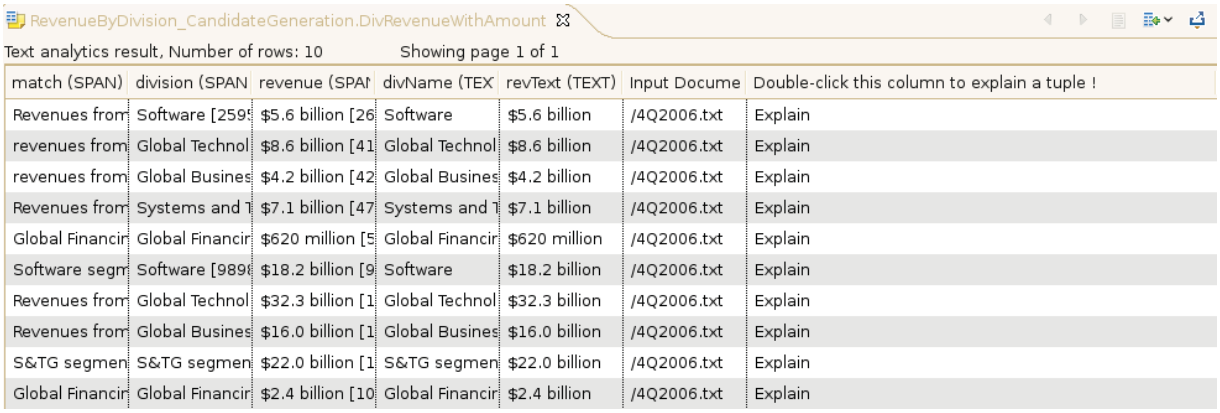

Figure 54 – Output View Table

8.2 Filtering Annotation Explorer Rows

1. Click on the ***Select Filters for the Annotation Explorer*** drop down button (highlighted in the figure below), and select ***Show hide filter view***.

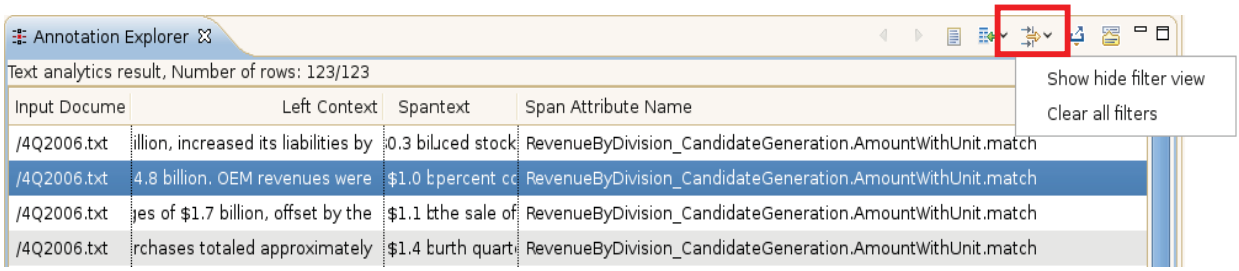

Figure 55 – Setting up filters for information displayed in Annotation Explorer

2. In the drop-down menu for *Filter Criteria* shown in below, check the boxes for '***Span Attribute Name Filter***' and '***Input Document Filter***' as shown in the figure below.

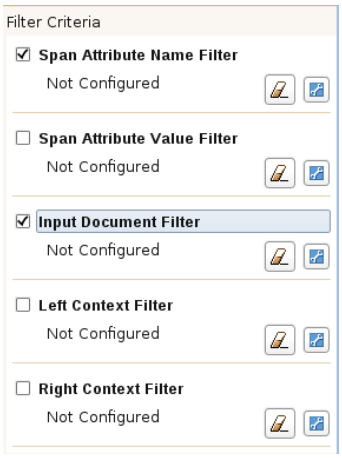

Figure 56 – *Selecting* the filter Criteria

Then click the wrench icon  to bring up the wizard for configuring the '***Span Attribute Name Filter***' shown in figure 57.

3. In the *Configuration Filter* wizard shown below, select the span name '**RevenueByDivision_CandidateGeneration.*DivRevenueWithAmount.match***' as shown, click ***Add*** and then click ***OK***. This sets up the permitted values of ***Span Attribute Name***, which is the *last* column in the Annotation Explorer view, for the rows displayed in Annotation Explorer.

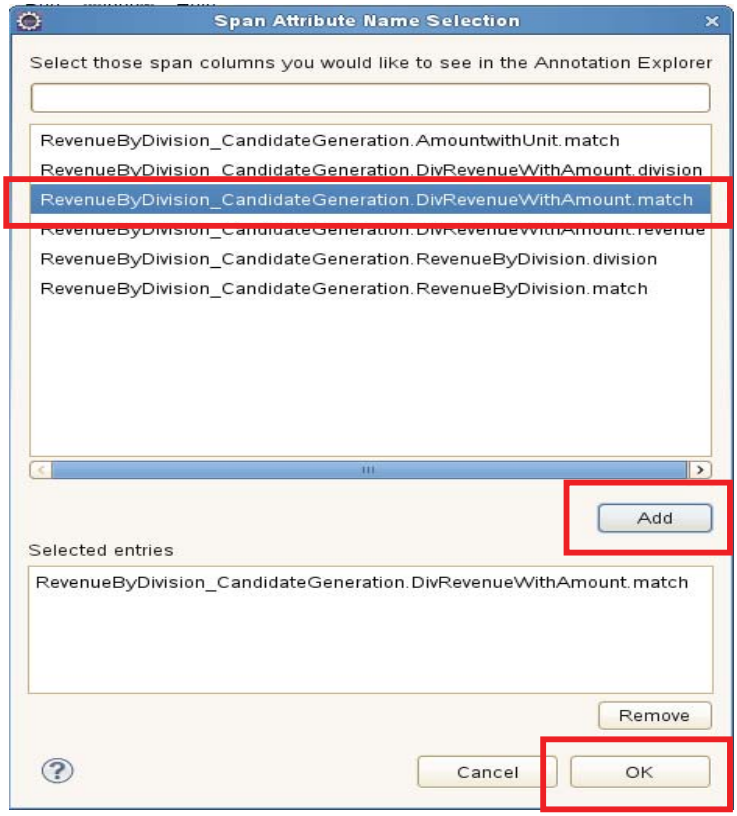

Figure 57 – Configuring each Filter Criteria

4. Next we click on the wrench icon  for *'**Input Document Filter**'* and add ***/4Q2006*** to the selected entries, then click ***OK*** to exit. This sets up the permitted values of Input Document, which is the *first* column in the Annotation Explorer view. Once the configurations for the two filters are set, we will see the 10 annotations in annotation explorer, as shown below.

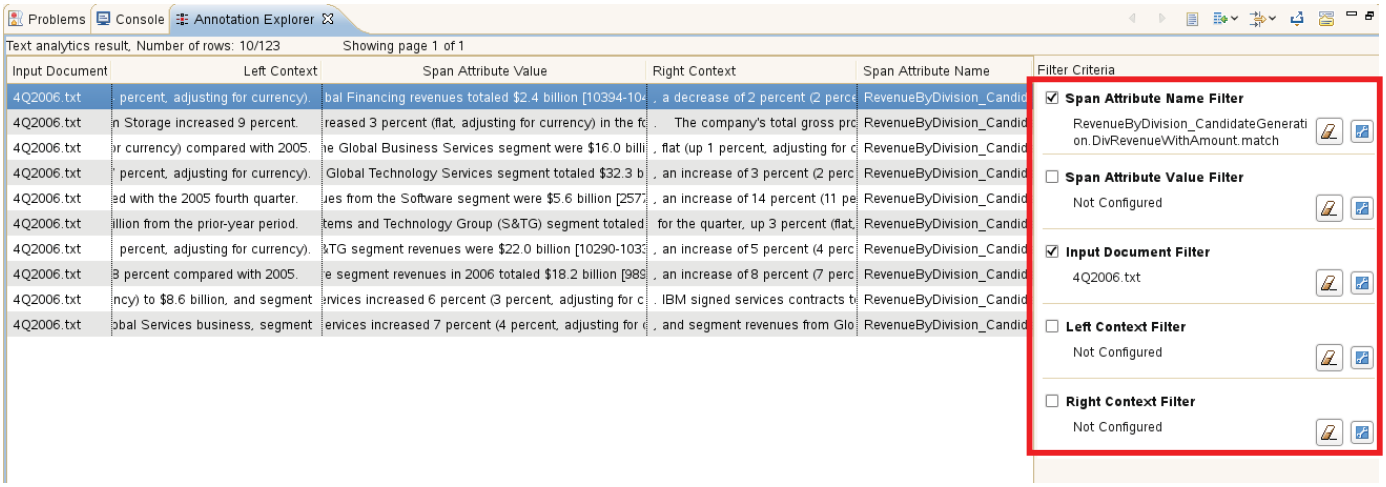

Figure 58 – Results of Filters applied – fewer rows displayed in Annotation Explorer, those matching the filter criteria

The filters for the Annotation Explorer view can be reset by clicking on the ***Clear all filters*** in the ***Select Filters for the Annotation Explorer*** drop down menu; this would reset the default view of Annotation Explorer. In general, one can choose any subset of filter criteria; and for each filter criteria, any subset of values.

5. By clicking on the filter icon again you can get rid of the drop down menu for Filter Criteria.

8.3 Exporting an Output View

1. A left-click on the '*Export Results*' icon highlighted in the figure below, will bring in the Export Results dialog box. Enter ***/home/biadmin/Desktop*** for the name of the output directory. Click ***Finish*** to close the dialog box.

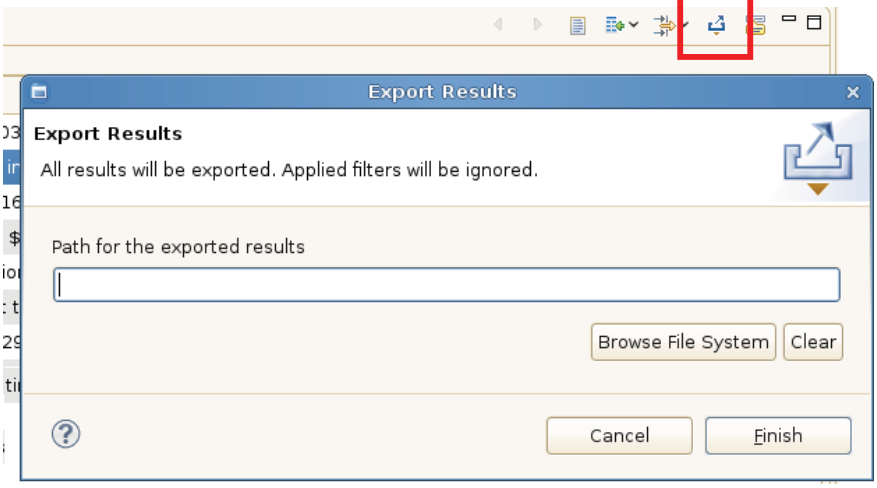

Figure 59 – *Exporting extracted views from Annotation Explorer tools bar*

2. Two folders named 'csv' and 'html' are created in the directory specified. Each folder contains a file for each output view. The ***DivRevenueWithAmount.html*** file is shown below .

Figure 60 – *Output html document for the view DivRevenueWithAmount*

8.4 Mouse-Over function to explain the annotated text

1. Left-clicking on the icon ***Disable Span Tooltip*** shown below toggles the mouse-over function on the annotated text document. When the mouse-over function is disabled, there is a red diagonal line over the icon as shown in the figure.

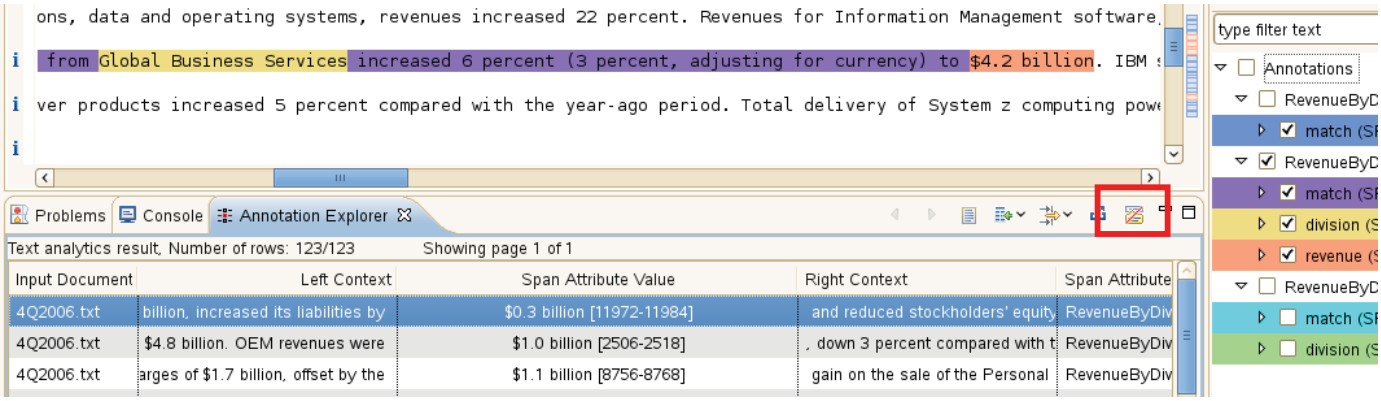

Figure 61 – *Clicking on the icon in red rectangle enables/disables the mouse-over function in Annotation Explorer*

2. When the mouse-over function is enabled, moving the mouse over an entry in Annotation Explorer view causes a pop-up. The pop-up contains the name of the top level view for the annotated text, the annotated text, and the span for the annotated text.

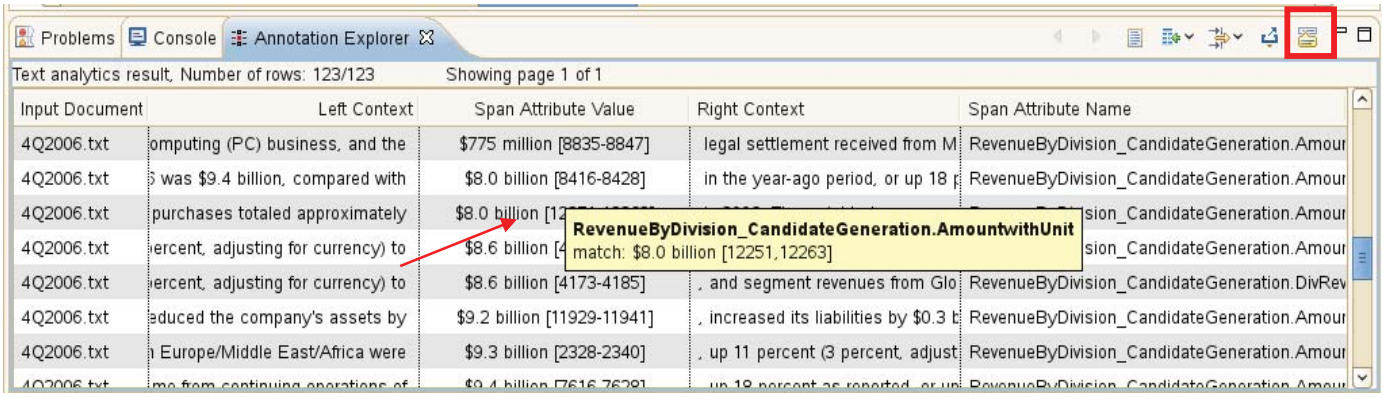

Figure 62 – *Illustration of mouse-over function*

9. Summary

This lab explored how to use the Text Analytics feature of BigInsights Enterprise. Working mainly with Eclipse with the BigInsights Eclipse Tooling plug-in installed, you were introduced to the ideas of basic features, concepts, patterns, regular expressions, and more. Following the steps of the Extraction Task view simplified the task of developing AQL to extract the required information. The Extraction Plan view was useful to keep track of all basic features, concepts, candidate generation, and so on. You can combine what you have learned here with other Big Data tools such as BigSheets, JAQL, and Streams.

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada