Instructions for OntoGen 2.0

1. Introduction

OntoGen[1] is a system for *data-driven semi-automatic ontology construction.* Phrases "semi-automatic" and "data-driven" stand for:

- **Semi-Automatic** – The system is an interactive tool that aids the user during the ontology construction process. The system suggests concepts, relations and their names, automatically assigns instances to concepts and provides a good overview of the ontology to the user trough concept browsing and visualization. At the same time the user can fully adjust all the properties of the ontology by manually adding or deleting concepts, relations and reassigning instances.
- **Data-Driven** – Most of the aid provided by the system (concept, relation suggestion, etc.) is based on some underlying data provided by the user at the beginning of the ontology construction. The data reflects the domain for which the user is building an ontology. Instance and instance co-occurrences are extracted from the data together with their profiles. Representation of profiles will be discussed later.

This document describes main parts of the system, their role in the learning process and how to use them. OntoGen is based on *TextGarden*[2] text mining software library and was developed under SEKT project. This document is mostly based on SEKT deliverable *D1-7-1 Ontology generation from scratch* [Fortuna05a] and *D1-12-1 Optimized ontology generation from scratch* [Fortuna06b].

2. Overview

The functionality of the system (machine learning algorithms, data handling and ontology visualization) is based on the *Text Garden library* [Grobelnik06] and is efficiently implemented in *C++*. The user interface is written in *C#* and requires *Microsoft .NET framework 2.0*[3] for running. The heavy-duty functionality is now compiled in native code and stored in DLL library. The library is accessed from the C# and used to provide functionality to the user interfaces. Previous version used *Managed C++* so the whole program was compiled into bytecode. By compiling the core functionality directly into native code the machine learning algorithms used in OntoGen run approximately twice as fast as in the previous version.

3. Detailed description

The main window (Figure 1) is divided into three main areas. The largest part of the windows is dedicated to *ontology visualization* and *document management* part (the right side of the window). On the upper left side is the *concept tree* showing all the concepts from ontology and on the bottom left side is the area where the user can check details and manage properties of the selected concept and get suggestions for its sub-concepts.

[1] OntoGen – `http://ontogen.ijs.si/`
[2] TextGarden – `http://www.textmining.net/`
[3] .NET 2.0 – `http://msdn.microsoft.com/netframework/`

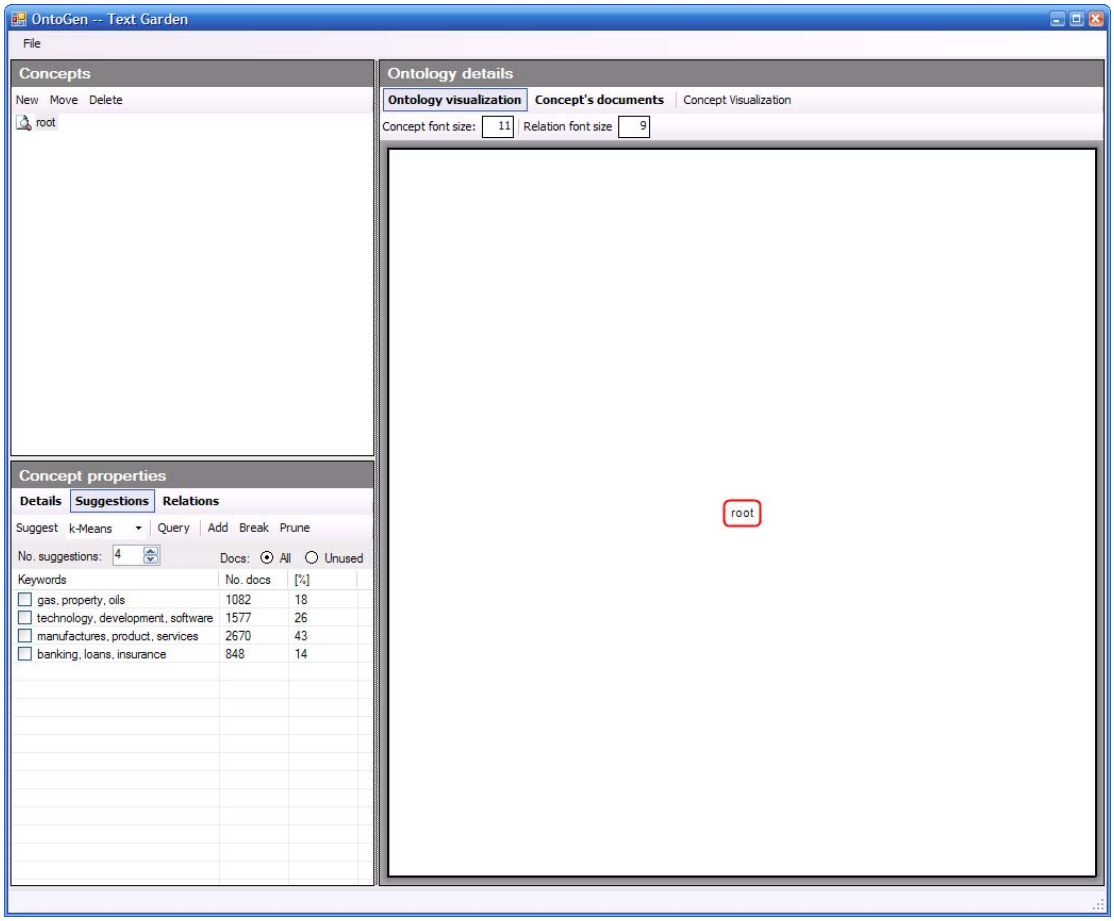

Figure 1

The arrangement of the main window is similar to the previous version with a few improvements mainly based on the suggestions we got from the users. Ontology visualization and list of concept documents now share the same window area and the user can switch between them. This gives him/her a larger portion of the window when they are in use which in turn makes visualization and document management easier.

3.1. Creating and saving ontologies

OntoGen supports several input formats for text instances (*Folder* and *Named Line-Documents*) and support for proprietary Text Garden format *Bag-Of-Words* as shown on Figure 2. Option *Folder* loads a set of HTML or Text files stored in a given file folder and option *Named Line-Documents* loads documents from a given file where each line is interpreted as a document with the first word in the line being its title.

If the instances already have assigned some preliminary labels in the input data, then OntoGen automatically asks if it should apply SVM word weighting method [Fortuna06a]. TFIDF word weighting is used by default.

Ontologies created in OntoGen can be saved as Proton Topic Ontology (also available in the previous version), RDF Schema or OWL ontology.

OntoGen is also integrated into OntoStudio as a plug-in. The user can use it for creating initial version of ontology which he can then further refine inside OntoStudio.

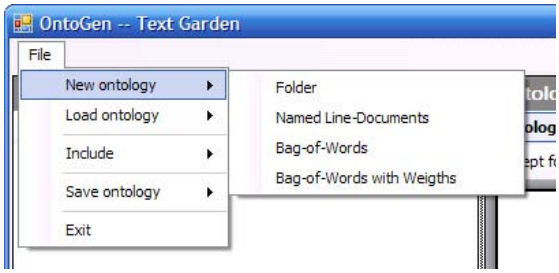

Figure 2

3.2. Basic concept management

The upper left side of the window offers a place where the user can get a quick overview of all the concepts and how they are position in the concept hierarchy (left side of Figure 3). The selections here and on the ontology visualization are synchronized.

Here the user can create a new empty concept (instances must be manually added to the concept by the user), reposition an existing concept to different position in the hierarchy or delete an existing concept from the ontology. Each of these tasks is achieved by first selecting the concept in the concept tree and then clicking the button that corresponds to the desired action: *"New"*, *"Move"* or *"Delete"*. When moving the concept, the program pops up a dialog window asking the user to select the destination concept (right side of Figure 3).

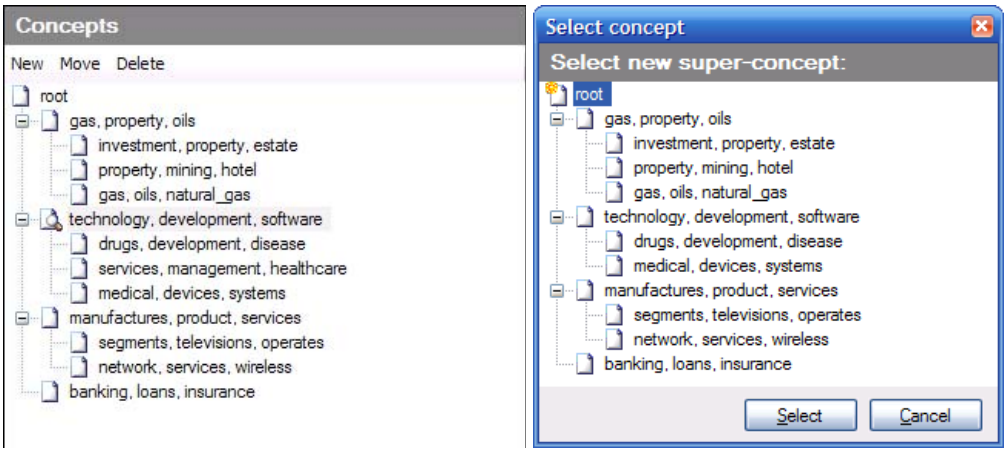

Figure 3

In the bottom left part of the main window (Figure 4) the user can edit the name of the selected concept and check the concept's main keywords. There are two keyword extraction methods implemented in the system and both of them are presented. The first one, shown under *Keywords*, is extracted from the concept's centroid vector and the second one, shown under *SVM Keyword*, is extracted from the concept's SVM linear model. SVM keywords are expensive to extract and are calculated only on user's request (with button *"Calc"*).

Information about the number of instances in the concept (*All documents*), number of instances not in any of the sub-concepts (*Unused documents*) and average inner-cluster similarity measure (*Avg. Similarity*) are also presented here.

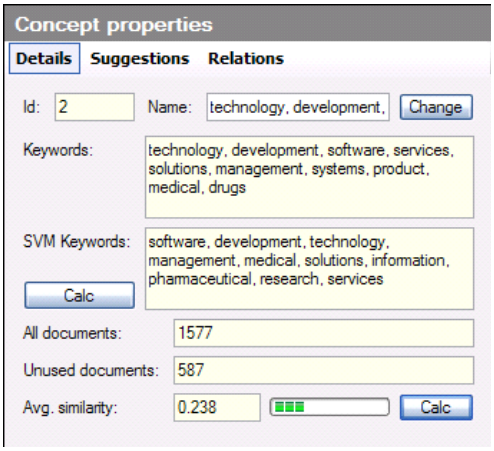

Figure 4

3.3. Concept suggestion

One of the main parts of the system is concept learning. In OntoGen this functionality is accessible under *"Concept – suggestion"* tab that can be seen in Figure 5.

There are two different approaches implemented for concept learning. In the *unsupervised* approach the system provides suggestions for possible sub-concepts of the selected concept. In the *supervised* approach the user has an initial idea of what a sub-concept should be about and enters it into the system as a query.

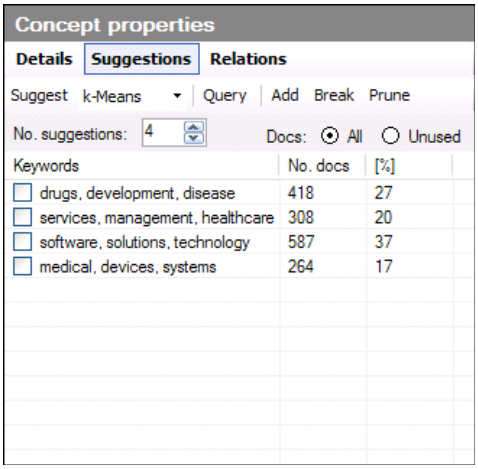

Figure 5

3.3.1. Unsupervised

There are three clustering methods implemented in OntoGen: k-means, LSI and PH k-means. The first two were already part of the first version while the last method is newly added in the latest version. There is also a suggestion method called *category* which just groups the instances according to the labels included in the input data. The user can select which method the system should use for generating suggestions.

The user can supervise the parameters for the methods (number of clusters or minimal inner-cluster similarity) and on which documents should the clustering be performed (*All* for all

documents in the concept and *Unused* for documents that are not already in any of the concept's sub-concepts).

After selecting the method and parameters the user can get sub-concept suggestions by clicking *Suggest* button. The clustering algorithm prepares suggestions and the program displays them in the list (see lower half of Figure 5). Each suggested sub-concept is described by the extracted main keywords (using the centroid method) and the size. A longer list of keywords is displayed when the user moves with mouse over the suggestion.

There are three follow-up tasks that the user can do with the suggestions. He can check the suggestions which s/he likes and add them to the ontology by clicking *Add* button. The suggestions are added as sub-concepts of the selected concept. He can replace the selected concept with suggested concepts by clicking *Break* button. This action removes the selected concept from the ontology and replaces it with the checked suggested concepts. All relations of the selected concept are redirected to/from the new concepts. Sometimes the system identifies a sub-concept for which the user thinks that should not be part of the concept. The user can decide to prune the suggested sub-concept (*Prune* button) from the selected concept which effectively removes suggested sub-concept's instances from the selected concept.

3.3.2. Supervised

A new feature in OntoGen is a supervised method for adding concepts. It is based on SVM active learning method. The querying and active learning is only applied to the instances from the selected concept (according to the parameters *All* or *Unused*).

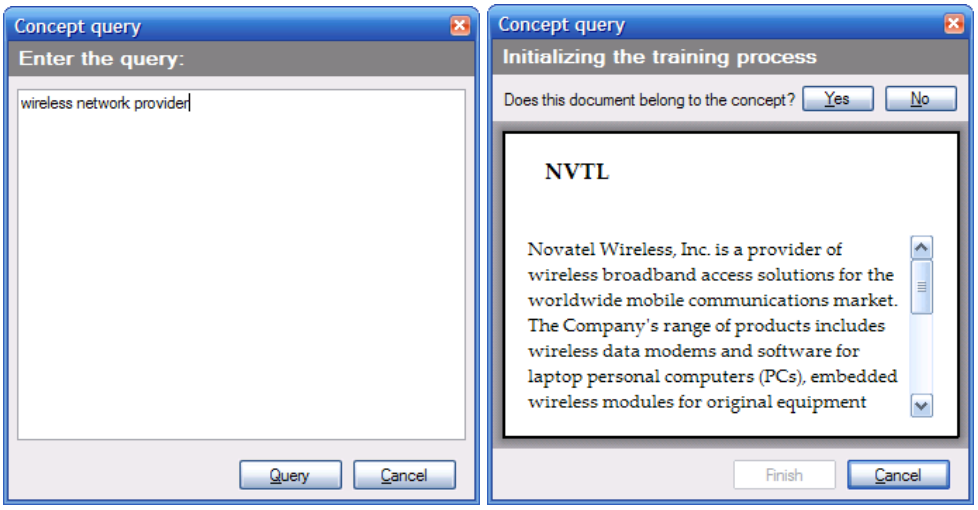

Figure 6

The user can start this method by clicking "Query" button in Figure 5. The system then launches a dialog that takes the query from the user (left in Figure 6). After the user enters a query the active learning system starts asking questions and labelling the instances (right in Figure 6). On each step the system asks if a particular instance belongs to the concept and the user can select *Yes* or *No*.

Questions are selected so that the most information about the desired concept is retrieved from the user. After some initial labelled sample is collected from the user the system displays some additional information about the concept (Figure 7). It displays the current size (number of

documents positively classified into the concept) and most important keywords for the concept (using SVM keyword extraction). The user can continue answering the questions or finish by clicking on the *Finish* button. The more questions that the user answers the more correct assignment of instances in the final concept are.

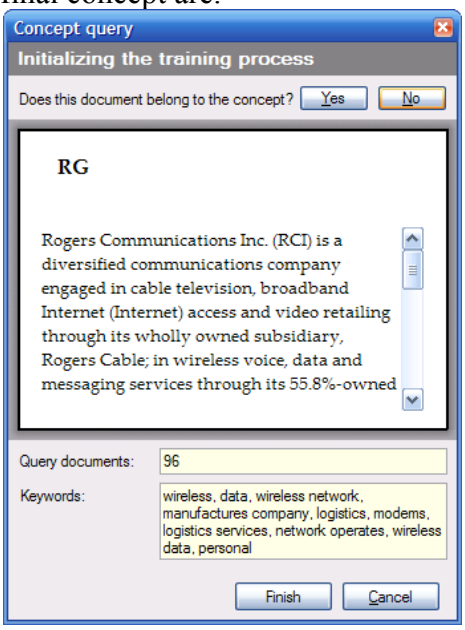

Figure 7

After the concept is constructed it is added to the ontology as a sub-concept of the selected concept.

3.3.3. Example of unsupervised and supervised concept suggestion

There is a fundamental difference between the unsupervised and supervised methods. The main advantage of unsupervised methods is that it requires very little input from the user. The unsupervised methods provide well balanced suggestions for sub-concepts based on the instances and are also good for exploring the data.

The supervised method on the other hand requires more input. The user has to first figure out what should the sub-concept be, he has to describe the sub-concept trough a query and go through the sequence of questions to clarify the query. This is intended for the cases where the user has a clear idea of the sub-concept he wants to add to the ontology but the unsupervised methods do not discover it.

3.4. Ontology visualization

While the user edits the ontology it is visualized in real time on the right side of the main window (Figure 1). The root concept is displayed in the centre of the visualization; sub-concepts of the root concept are displayed in the circle around the root concept, and so on. The user can select a concept by clicking on it on the visualization. The currently selected concept is drawn in red colour, other concepts are drawn in blue colour and the relations are drawn with green colour (see Figure 8 for example of visualization).

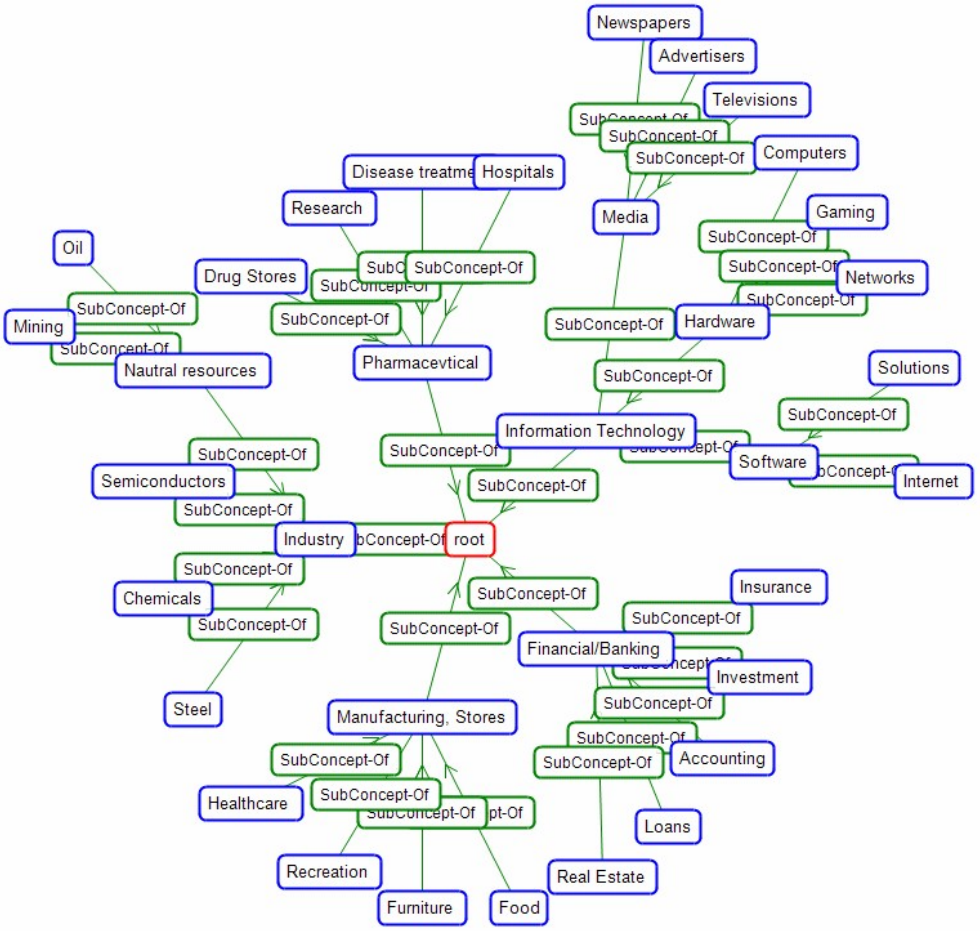

Figure 8

3.5. Concept document management

After a new concept is added to the ontology, either with unsupervised or supervised methods, the system automatically assigns instances to it. Since this assignment is often not perfect the user can manually reassign instances to and from the concept.

This functionality is provided through the interface shown in Figure 9. On the left side is a list of all the instances. The instances from the concept are checked. Assignment of instances can be changed by checking or un-checking them. The user must click *Apply* button at the end to confirm the changes. The content of an instance can be seen by selecting it from the list – it is displayed on the right side.

The system calculates similarity between each instance in the list and the centroid vector of the concept (*cosine similarity* is used). The instances can be sorted according to similarity. At the bottom of the screen is a similarity graph with instances sorted according to similarity on the *x* axis and similarity on the *y* axis. Instances from the concept are shown in red colour. Similarity can be very practical when searching for outliers inside the concepts or for the instances that are not in the concepts but should be considering their content. User can quickly add or remove instances from the concepts by simply selecting them on the graph with the mouse.

OntoGen also has functionality for detecting specific inconsistencies inside ontology. When an instance does not belong to a concept but it is in one of its sub-concepts, then it has red background in the list on all instances (for example check Figure 9). The user can, based on this information, assign the instance to the concept or remove it from its sub-concepts by right clicking on the instance and selecting *Remove from sub-concepts* from the menu.

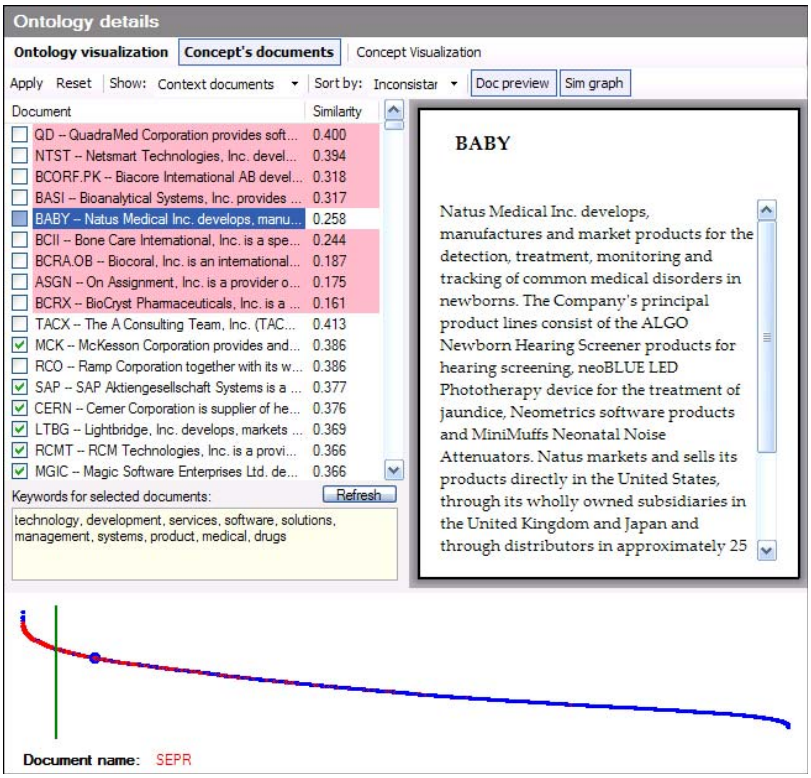

Figure 9

3.6. Concept visualization

Instances of selected concept can be visualized by clicking *Concept Visualization* button. Document Atlas tool is used for visualization, more details about it can be found in [Fortuna05b].

3.7. Relation management

The user can also edit other types of relations besides the "sub-concept of" relation. This is done trough relation management screen shown in Figure 10. Here the user can see a list of all relations of the selected concept. He can add new relation, delete existing ones or add new types of relations.

Figure 10

New instance importing

The new version of OntoGen also enables the user to add new instances to an existing ontology. The procedure is started by selecting a source of new instances in the menu *File→Include*. The system can load new instances from *Folder* or *Named Line-Documents*. Then the system trains SVM classifiers on the instances already arranged into ontology and uses them to classify new instances (Figure 11).

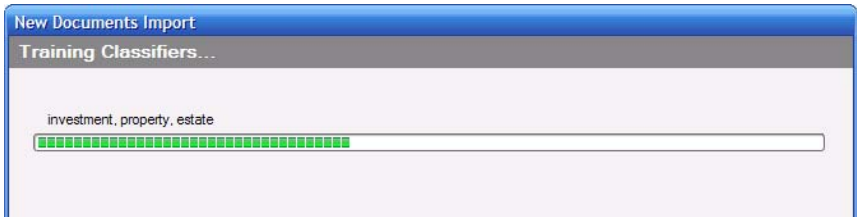

Figure 11

In the next step the OntoGen presents to the user a list of all the newly imported instances and their classification results (Figure 12). User can check and correct classifications for each of the instances by first selecting the instance from the list and then checking the appropriate concepts in the concept tree. Preview of the selected instance is also displayed to aid the user. The instances are automatically added to the ontology after the user clicks *Finish*.

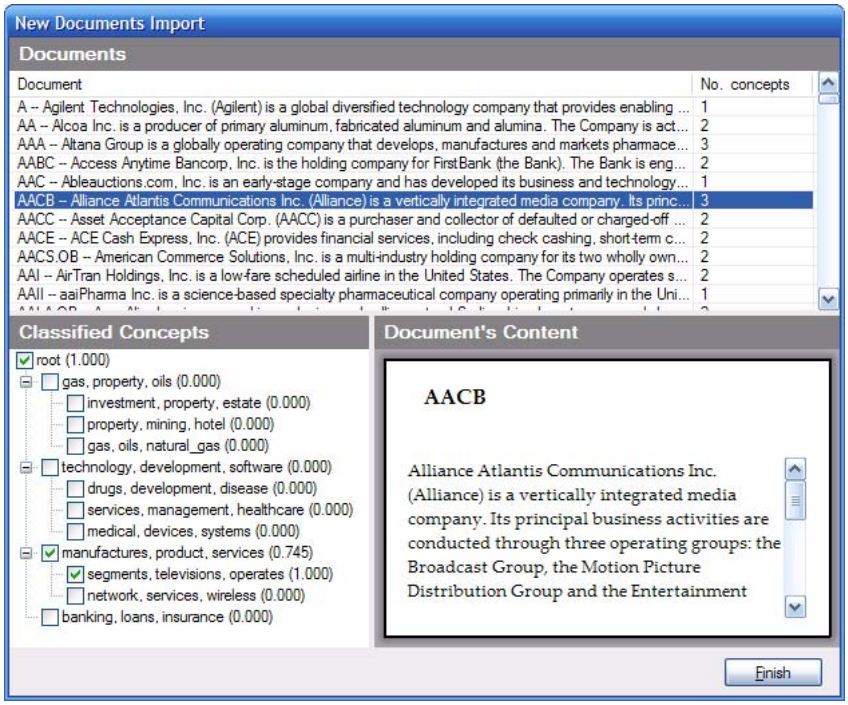

Figure 12

Acknowledgment

This work was supported by the Slovenian Research Agency and the IST Programme of the European Community under SEKT Semantically Enabled Knowledge Technologies (IST-1-506826-IP) and PASCAL Network of Excellence (IST-2002-506778). This publication only reflects the authors' views

References

[Fortuna05a] Fortuna, B., Grobelnik, M. Mladenić, D. *D1.7.1 Ontology generation from scratch*. SEKT Deliverable.

[Fortuna05b] Fortuna, B., Grobelnik, M., Mladenić, D. *Visualization of Text Document Corpus*. Informatica 29 (2005), 497-502.

[Fortuna06a] Fortuna, B., Grobelnik, M. Mladenić, D. *Background Knowledge for Ontology Construction*. WWW 2006, May 23.26, 2006, Edinburgh, Scotland

[Fortuna06b] Fortuna, B., Grobelnik, M. Mladenić, D. *D1.12.1 Optimized ontology generation from scratch*. SEKT Deliverable.

[Grobelnik06] M. Grobelnik, D. Mladenic. *Text Mining Recipes*, Springer-Verlag, Berlin; Heidelberg; New York (to appear), 2006, (accompanying software available at `http://www.textmining.net`).