H-0289 (H1008-007) August 9, 2010
Computer Science

IBM Research Report

Enterprise Data Classification using Semantic Web Technologies

David Ben-David
Technion - Israel Institue of Technology
Haifa 32000, Israel

Tamar Domany, Abigail Tarem
IBM Research Division
Haifa Research Laboratory
Mt. Carmel 31905
Haifa, Israel

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Enterprise Data Classification using Semantic Web Technologies

David Ben-David[1], Tamar Domany[2], and Abigail Tarem[2]

[1] Technion – Israel Institute of Technology, Haifa 32000, Israel,
davidbd@cs.technion.ac.il
[2] IBM Research – Haifa, University Campus, Haifa 31905, Israel,
{tamar,abigailt}@il.ibm.com

Abstract. Organizations today collect and store large amounts of data in various formats and locations. However they are sometimes required to locate all instances of a certain type of data. Good data classification allows marking enterprise data in a way that enables quick and efficient retrieval of information when needed. We introduce a generic, automatic classification method that exploits Semantic Web technologies to assist in several phases in the classification process; defining the classification requirements, performing the classification and representing the results. Using Semantic Web technologies enables flexible and extensible configuration, centralized management and uniform results. This approach creates general and maintainable classifications, and enables applying semantic queries, rule languages and inference on the results.

Keywords: Semantic Techniques, RDF, Classification, modeling

1 Introduction

Organizations today collect and store large amounts of data in various formats and locations. The data is then consumed in many places, sometimes copied or cached several times, causing valuable and sensitive business information to be scattered across many enterprise data stores. When an organization is required to meet certain legal or regulatory requirements, for instance to comply with regulations or perform discovery during civil litigation, it becomes necessary to find all the places where the required data is located. For example, if in order to comply with a privacy regulation an organization is required to mask all Social Security Numbers (SSN) when delivering personal information to unauthorized entities, all the occurrences of SSN must be found. That is when classification becomes essential.

Data discovery and classification is about finding and marking enterprise data in a way that enables quick and efficient retrieval of the relevant information when needed. Since classifying and tagging everything in the organization's data sources is time-consuming, inefficient and rarely done well, usually organizations need to choose what to classify, determining the relevant and important pieces of information that need tracking. The classical approach today is to start by

examining the purpose of the classification and how the classified data is going to be used. Examples may include the need to comply with privacy regulations, data retention or archiving policies. The process consists in manually examining the relevant laws or regulations, identifying which types of information should be considered sensitive and what are the different sensitivity levels, and then building the classes and classification policy accordingly. The main drawback of such an approach is that each time the requirements change (e.g., due to changes in existing laws or internal policies or the addition of new laws) it requires re-classification of the data.

Moreover, enterprise data can be stored in many varied formats: unstructured data (such as word processing documents, e-mails or blobs in databases), semi-structured data (data that has a machine processable structure but is not fully structured, such as XML, HTML, and RDF/OWL) and structured data (well formatted, with a well defined structure, usually relational datasets); and is typically stored in various data stores such as relational databases, file systems, mail servers, etc., sometimes even at different physical locations (sites) in the organization. Most existing solutions for automatic data classification apply to a single data type, mainly unstructured data, and can generally only identify predefined sets of known fields, depending on the domain.

The need to build a generic classification system with manageable results introduces several challenges. First, a common 'language' that can be used as input to all classifiers must be devised. This 'language' must enable an accurate classification process but also provide flexibility and extensibility to new types of classifications, classes and data types. In addition, a uniform format for the classification results must be created so as to maximize their usability and homogeneity.

In this paper we introduce a generic, automatic classification method that enables classifying data elements from all types, formats and sources based on a common classification scheme, and making it possible to use the results in a uniform manner. We suggest a method based on Semantic Web technologies for classifying any type of data from any domain, by modeling the information in question as an ontology. According to our approach, a model (or ontology) provides the means for describing the entities to discover, the classes to map them to, and representing the results of the classification. In addition to describing what entities to discover, it also contains information on how to discover them. Such an ontology can serve as input to a classifier, which uses the information depicted in it to discover the relevant data elements in the data source. The same ontology can serve as input to several classifiers, each for a different source of data. The ontology is then used again as a schema for representing the discovery results, thus unifying the results across the different sources.

The ontology used as input for the classification process can describe any type of content. It can be generic, containing information that may apply to many different organizations and/or requirements, such as knowledge on what identifies a person (PII); it can be specific to a certain domain, such as medi-

cations and treatments for the Healthcare domain; and it can be specific to a certain organization or even a particular application.

We use the Resource Description Framework (RDF) and its extensions (RDFS and OWL) to represent the classification models and results. By using RDF for these purposes we maximize modularity and extensibility and facilitate easy navigation between the results, the data models and the actual data in the data sources, as will be described in the following sections.

Using models, in combination with RDF and additional Semantic Web tools gives us several advantages. First, it enables decoupling the classification process from the intended usage, allowing more general and maintainable classifications with less dependency on changes. Using the model as a common input to all classifiers enables centralized management, auditing and change control. A uniform format for all classification results enables fast and easy location of the data when needed, enabling organizations to annotate their data with semantic meaning, thus increasing its usability. It is then possible to apply semantic queries on the results using query languages such as SPARQL [19], feed the results to inference or reasoning engines to deduce additional relations, and even apply rule languages such as AIR [12] and TAMI [28] to enforce and verify compliance with existing policies.

The paper is organized as follows: Section 2 describes related work, both in the area of classification in general, and in the use of RDF in information management. Section 3 describes the general idea of classification models, as well as the PII model as a specific example. In Section 4 we discuss general requirements from classifiers and describe in detail a relational data classifier. In Section 5 we give a short description of our implementation and some results, and we conclude in Section 6.

2 Related work

2.1 Classification

There are many existing algorithms for automatic classification of unstructured documents, using various techniqes such as pattern maching, keywords, document similarity and different machine learning techniques (Bayesian, Decision Trees, Resource Limited Immune Classifier System, k-Nearest Neighbor and more), as well as technologies such as Autonomy's Meaning-Based Computing Technology [2]. This paper does not suggest any specific algorithm, and any of the above can be used as part of the classification process.

In many existing works, the classes and the entities to discover are predefined, deriving from the specific use case and domain of the solution. There are several works that suggest a way to find and describe the classes. Miler [18] suggests a method for automatically enriching the classes based on the discovery. Ben-Dor et al. [3] suggest a way to find the meaningful classes from the content in the computational biology domain. Taghva at al. [24] suggest to use ontologies to describe the classes when classifying emails.

Our approach is to enable modelling any type of data to be classified, from any domain, and to use the same model to describe the classification schema for all input types, thus simplifying the creation and management of classification policies across the enterprise.

Most work related to classifying relational data deals with classifying or clustering records based on common patterns or associations. Much less has been done in the area of classifying based on metadata (tables, columns, types), according to specific categories. The same applies to semi-structured data such as XML files. One scenario in which we found some similar work is electronic discovery (eDiscovery)[3]. Butler et al. [6] developed a top level ontology to represent enterprise data for eDiscovery, and created a semantic mapping that manually maps the data models of heterogeneous sources into the ontology.

We suggest a method to automaticaly classify data from different sources according to a model, which is both flexible and extensible, to enable accurate classification at the level of a single data element.

2.2 Use of ontologies and RDF in Information Management

The use of Semantic Web methodologies in the world of information management, has lately gained much momentum.

The use of ontologies in information and knowledge management has long been recognized as a natural connection. According to Staab at al. [23], ontologies open the way to move from a document-oriented view of Knowledge Management to a content-oriented view, where knowledge items are interlinked, combined, and used. They present an approach for ontology-based Knowledge Management as a methodology for developing ontology-based Knowledge Management systems. Maedche et al. [17] also present an integrated enterprise-knowledge management architecture, focusing on how to support multiple ontologies and manage ontology evolution. Ontology-driven information and knowledge management in specific domains has also been investigated [8].

RDF specifically can also be used to unify data access across multiple sources. Langegger et al. in their work [16] presented a system that provides access to distributed data sources using Semantic Web technology designed for data sharing and scientific collaboration. Warren and Davies [27] studied a semantic approach to information management for integrating heterogeneous information as well as coping with unstructured information.

RDF has recently begun to be used in the context of data classification. Jenkins et al. [10], use RDF to express the results of classifying HTML documents using the Dewey Decimal Classification for automatically extracting context sensitive metadata. In addition, Xiaoyue and Rujiang [29] showed how using RDF ontologies to perform concept-based classification of text documents, as opposed to the traditional "Bag of Words" approach, significantly improved text classification performance.

[3] Discovery in civil litigation that deals with information in electronic format

We use RDF to express both the input and the output of the classification process, thus enabling high connectivity and easy navigation between the models, the results and the original data sources and providing a unified representation of all classified data sources.

2.3 Existing models

Models are widely used to describe domain-specific concepts. IBM's Industry Models[4], combine industry knowledge and best practice in a usable form for both business and IT communities in order to accelerate industry solutions. The Dublin Core Metadata Initiative (DCMI)[5] is an open organization engaged in the development of interoperable metadata standards that support a broad range of purposes.

The Friend of a Friend (FOAF) [14] project's goal is to make it easier to share and use information about people and their activities on the Web and to transfer information between Web sites.

3 Classification models

The main goal of our solution is to enable organizations to annotate their knowledge bases with semantic meaning. Our approach offers to automate the classification process using classification models, which are ontologies that describe entities in the organization's knowledge base and their relationships. These models can be generic, domain-specific or even organization-specific.

The classification process is performed at the single data element level and results in bi-directional references between the individual data parcels and terms in the model. This method enables the "classify once, enforce many" approach for future application of policies and other semantic applications.

The Data Discovery and Classification process involves four steps:

1. Defining and identifying the data to discover (the valuable pieces of information to locate).
2. Building a classification scheme. One example of a common classification scheme is top secret, secret, confidential, etc.
3. Discovering where the valuable data is located.
4. Documenting the findings in a useful manner.

We offer to use the same model for several purposes throughout the discovery and classification process: the description of what entities to search for (step 1), the classification scheme to which the findings are associated (step 2), directives on how to search for each entity (input to step 3) and the schema for representing the discovery results (step 4).

[4] `http://www-01.ibm.com/software/data/industry-models`
[5] http://dublincore.org/

Selecting RDF as the language in which to represent the classification model enables us to exploit existing and evolving tools for annotating, reasoning, querying, etc. the classified data. We can take advantage of common vocabularies (RDFS, OWL) to express stronger relations and properties, best suited for ontologies, and benefit from the powerful features of RDF that makes it easy to expand, merge and combine existing classification models, and generate new models for different purposes.

3.1 Models as input

The classification models are used as input to an automatic classification tool (henceforth: *Classifier*), which is the software component that performs the actual mapping. Our goal is to have a general-purpose classifier, capable of handling any domain-specific knowledge, and so the input for the classifier must be designed to answer not only 'what to search for', but also 'how to search'. This means that in addition to functioning as an ontology, the classification model must also contain additional characteristics of the entities such as type, format and possible synonyms. This allows full automation of the classification process by use of natural language processing and information retrieval techniques.

To implement such a general-purpose classifier, a need emerged for a meta-model to describe the expected structure and contents of a valid classification model. An input model complying with this meta-model is regarded as an instance of the meta-model.

3.2 The Meta-model

We designed a preliminary meta-model for classification models used as input to our system implementation, which is depicted in Figure 1. The top-most element in the classification meta-model is the subject element that is the root node of any domain-specific ontology. A subject can possibly contain several categories, and each category can hold several fields, which are the basic terms linked by the classifier to data units in the organization's data sources. Each field can be associated with a data type and format (e.g., denoted by a regular expression), and one or more synonym elements, which represent keywords or phrases that can be used to describe the field. Each keyword (or token) in a synonym can be stated as optional, required or exact. These synonyms are processed by the classifier to create different search strings used to search for potential matching elements in the data stores. We elaborate on this matter in section 4.2.

The meta-model also supports mapping between a field in the classification model and one or more actual data elements. This mapping, obtained as the result of the classification process, is also represented in the model.

3.3 Models as output

The classification result is also saved in RDF format, as an extension of the models used to perform the search. It is composed of triples, each triple representing a data field that was discovered as belonging to one of the classification fields.

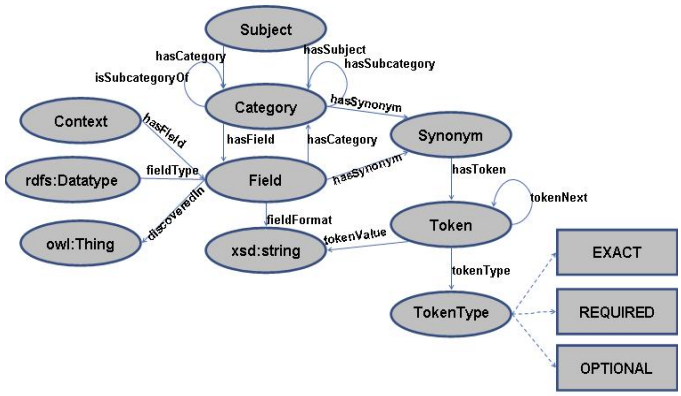

Fig. 1. The classification meta-model

The advantages in using RDF to represent the classification results are many:

1. Different results from different runs can be linked together to form one unified classification source containing all discovered entities (a concept demonstrated by Langegger at al. in [16]). If an existing model has been extended or a new model introduced, it is possible to perform a partial search only on the new parts, and combine the new results with the existing ones. Using RDF it is possible to express information on the location of the classified data (such as a database column or XPath), as well as the data itself (an actual value).
2. Thanks to the use of URIs to describe the different resources, it is very easy to navigate between the discovery results, the classification models and the RDF representations of the searched data sources. In this way it is possible, while accessing the results, to access additional information about the model or data source to verify that the results are correct, add new links or add new entities to the models to improve future discovery. It is also possible to obtain additional information about certain resources from external knowledge-bases.
3. The use of RDF allows performing semantic queries on the results. Many existing query languages for the RDF language (such as SPARQL [19], RDQL [20] and many more) can be utilized to extract useful information from the classification results. This information may be used to verify the correctness of the results or even learn new insights that can be used to further enrich the models and/or the search algorithms to improve the discovery process.
4. Inference (or reasoning) can be applied to the existing RDF triples to derive additional RDF assertions that are not explicitly stated. These new assertions are inferred from the base RDF together with any optional ontology information, axioms or rules. Many existing inference engines or reasoners include a generic rule engine that receives a set of base asserions or axioms

(which serve as rules), and can process any input RDF instance to infer new statements base on those rules. There have also been a few attempts to bridge the gap between RDF and logic programming and languages [5]. N3logic [25] provides a logical framework to express logic in RDF. It enables the use of rules to make inferences, choose courses of action, and answer questions, and constitues a minimal extension to the RDF data model such that the same language can be used for logic and data.

5. Several policy and rule expression languages for RDF (such as AIR [12], KAoS [26], and Rei policy language [11], etc.) and tools for checking compliance with policies (such as TAMI [28]) exist, enabling the enforcement of privacy rules or any other business policies on the discovered data.

3.4 Example: PII model

As a sample use case, we may consider the field of information security. In modern times, organizations retain an explosively growing volume of sensitive data to which access is available to an expanding user base. The growing public concern over the security and privacy of personal data has placed these organizations in the spotlight, and countries around the world are developing laws and regulations designed to support confidentiality.

The definition of what constitutes private information is subject to different cultural views. Personally Identifiable Information (PII) refers to information that can be used – whether alone or in combination with other personal or identifying information – to uniquely identify a single individual. For example, national identification numbers, vehicle registration plates and fingerprints are often used to distinguish individual identities and are clearly PII. Even less identifying characteristics such as city of residence, gender and age are considered PII for the potential that when joined together they might be linked with other public available information to identify an individual.

By successfully identifying and classifying PII in an organization's data, it is possible to apply virtually any privacy policy. Figure 2 displays a partial view of the PII model we have developed. The root element is Person, and it contains categories such as PersonName and Address. Each category contains fields such as firstName, middleName and lastName.

3.5 Advantages of classification using models

There are several advantages for using ontologies in general, and RDF in particular, in the discovery and classification process:

- **Extensibility** and **Flexibility**: The model can be extended both by adding more entities to search for and by adding additional tips and hints to aid the discovery of existing entities. Since the information on how to search for the entities is included in the model that is used as input to the discovery algorithm, the model can be extended, edited or replaced without the need to change the discovery algorithm itself.

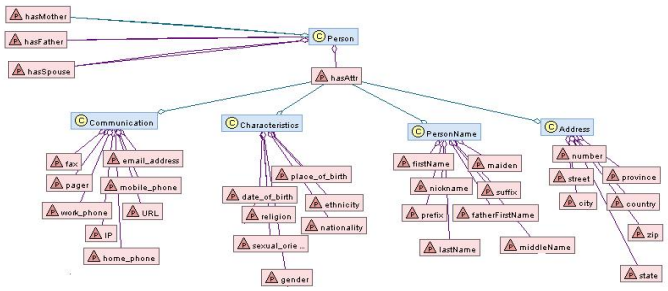

Fig. 2. Partial PII RDF Model

- **Linking to external knowledge bases**: The models used to perform the classification can themselves be derived from existing business models and/or domain- or industry-specific models and ontologies. Furthermore, additional knowledge from external ontologies or taxonomies can be linked into the model, enabling automatic extension of the model, addition of new synonyms, related terms, etc.
- **Resistance to policy changes**: By classifying data sources according to specific classes of information (first name, phone number...) as opposed to general, abstract categories (confidential, top-secret...) we can use the same classification to enforce several different policies and no reclassification is needed in case of policy changes.
- **Homogeneity**: By using a model containing the sensitive entities and classification schema as input, the same model can be used as input to many different classifiers that classify different types of data sources (unstructured documents, forms, emails, XML files, relational data, etc.). By using the same output format, we unify the classification results and improve their usability. For example, privacy policies can be enforced in a unified manner across all data types and formats in the organization.
- **Centralization**: Everything is linked through the use of unique IDs. This allows the model to serve as a centralized point to manage all valuable information in the organization and enables easy location of all related pieces of data in one click. On the other hand, given the unique ID of data location, we can easily find its classification.

4 Classifier

Once the models describing the data the organization wishes to classify are designed, these models are used as input to a software component that performs the classification of the actual data stores. This component is referred to as the *Classifier*. In this section we will describe the approach in general and a relational data classifier in more detail.

4.1 Classifier inputs

The classifier receives as input RDF files of two types: the first type is the RDF files representing the models described in the previous section; the second type is RDF files representing the data to be searched, for example, the schema of a database. The classifier performs a search for relevant items in the data schemas, refines these results according to several filters, and creates a new RDF file containing the search results.

Using one or more models to perform the search. As mentioned in previous sections, one of the main advantages of using RDF for representing the classification models is extensibility. Existing models can easily be extended by simply adding additional RDF triples. Additional models can also be created for domain or organization-specific classification. When performing the search on the different data sources, one or more models can be used as input, and in this way all data fields relevant to that specific search are found. Different searches may be performed on different combinations of models to yield results for different purposes such as privacy enforcement, retrieving specific information within a set timeframe in order to meet regulations, de-duplication and increasing data quality, etc. Each model used as input to the classifier must be an instance of the classification meta-model described in Section 3.2.

RDF representation of data source schemas. Organizations today have many different data sources, which may include structured data (e.g., relational databases), semi-structured data (e.g., XML files), and unstructured data (e.g., e-mail archives and text documents). RDF provides a means to represent different types of schemas in a unified manner in order to perform semantic queries on them. This is what allows us to perform the classification in a uniform manner on all data sources, without the need for any prior knowledge about the specific underlying schemas.

Much work has been done in the domain of representing relational database metadata in RDF format. RelationalOWL [15] is an OWL-based representation format for relational data and schema components. The D2RQ API [4] provides access to relational database content from within the Jena and Sesame RDF frameworks.

Some work has been done in the direction of transforming XML schemas and instances to RDF format. "RDF-enabled" XML files can be transformed to RDF using specific XSLT transformations, as described in [22] and as demonstrated in the XML2OWL[6] and the XSD2OWL and XML2RDF of the ReDeFer project of Rhizomik[7] initiative tools. However, a completely generic technique for fully representing any XML file in RDF has still not been found at the time of writing this article.

[6] `http://www.avt.rwth-aachen.de/AVT/index.php?id=524`
[7] http://rhizomik.net/html/redefer/

Regarding unstructured data, the search techniques must of course differ, since there is no underlying data schema. A great deal of work has been done on classifying unstructured data: several tools, such as SystemT [13], InfoSphere Classification Module (ICM)[8], RSA Data Loss Prevention Suite[9] and Symantec Data Loss Prevention product[10] analyze and extract information from unstructured text such as documents and e-mails. Many new methods for efficiently classifying unstructured text have been examined [21], and some domain-specific techniques have been developed.

We believe that classifying unstructured data based on classification models can greatly enhance the existing techniques for the classification of this type of data, since it enables a clear separation between the data classification phase and the policy enforcement phase. Moreover, representing the classification results in RDF format enables high connectivity between the search results on the different data types, and increases the variety of possible uses for these results, a concept that has also been used in [1].

4.2 Performing the search

The model used as input to the discovery process describes the entities for which to search and how to search for them. For each such entity there is a list of synonyms that describe the concept, and for each synonym we can indicate which words (or tokens) in the phrase are required, which are optional and which need to appear in the same exact form. These synonyms are used to create several search strings to compare to the data store's metadata. In the example of relational databases, they are compared to the database metadata, i.e the table and column names. For example, for the term "family name", the additional synonyms "second name" and "surname" can be stated, and for the synonym "second name" we can indicate that the word "second" is required but "name" is optional. In addition an entity can have one or more types associated with it and additional information on the expected format of the data. These are used at runtime to verify that a column suspected of containing a certain data element from the model indeed matches the expected type and format for that element.

During the discovery process, for each data element in the input models, a list of search strings is created from the synonyms defined for that element. The aim is to cover all common abbreviations and hyphenations that may be used to describe that data element in the data sources. The algorithm uses syllabifying techniques to divide each token (which is not required to appear exactly) into syllables, creates different possible substrings representing each syllable according to specific rules, and then uses all possible combinations of the substrings and tokens to search the metadata representation for matching data. In this phase, in addition to syllabification, additional linguistic techniques can be employed, such as stemming and affix stripping.

[8] http://www-01.ibm.com/software/data/content-management/classification

[9] http://www.rsa.com/node.aspx?id=3426

[10] http://www.symantec.com/business/data-loss-prevention

5 Implementation

We developed a reference implementation for model based discovery and classification in relational databases. We used the PII model described above as our test case, but any model that complies with the meta-model can be used just as easily. We chose the RelationalOWL ontology with some slight changes to describe database metadata in RDF, and used their implementation to extract the metadata from the database to RDF format. The current implementation uses the syllabifying techniques described in Section 4.2, as well as type checking. Future extensions are planned to include the use of additional linguistic tools (such as stemming) and the use of sample content to verify the data's format.

5.1 NeOn-based implementation

As a basis for this implementation we chose to use the NeOn toolkit [9], which is an open-source ontology engineering environment based on the Eclipse platform. The toolkit provides an extensive set of plug-ins covering different aspects of ontology engineering. In addition we used the Eclipse Data Tools Platform (DTP)[11] for defining connections with local and remote databases, RelationalOWL [15] for creating an RDF representation of the database metadata, and the Jena framework [7] for accessing and querying the different RDF representations. We implemented several Eclipse plug-ins that perform the actual discovery, integrate between the different components in the system and contribute some elements to the NeOn user interface.

In the classification tool, a user can create projects, create and edit models, import existing models (in both cases, the models are validated against the meta-model) and import or create database metadata RDF representations. The discovery process can be performed on any combination of models and databases, and the results are also displayed in the project. The results (as well as the models and database metadata) can be viewed in both a hierarchical view and a graph view, as depicted in Figure 3.

Thanks to the use of RDF to represent the discovery results (reminder: each triple in the result represents a data field that was discovered as belonging to one of the classification fields), it is possible to navigate from any node in the result back to both the classification field in the model, and to the data field (column) in the database RDF representation. This easy navigation allows verifying the classification results, refining them (adding or removing triples), and enriching the model to be more accurate in subsequent runs. As all of the information created and used by the system (models, metadata RDF representations, results) is stored in an RDF store, it can also be exposed to additional existing and evolving Semantic Web tools that operate on RDF stores.

[11] `http://www.eclipse.org/datatools`

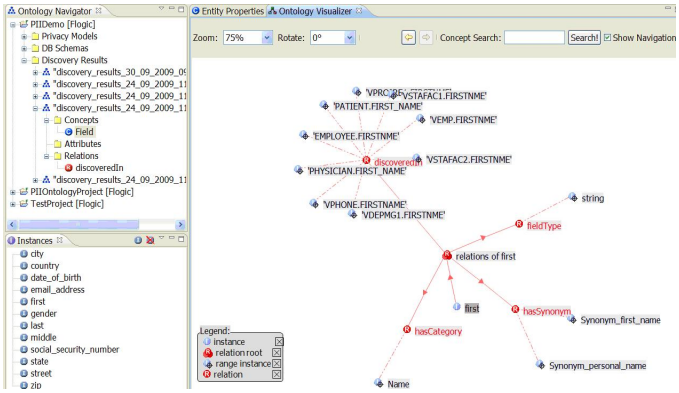

Fig. 3. A partial view of the discovery results in the NeOn-based tool

5.2 Some Results

We ran our discovery algorithm on two externally available databases: one representing employee records in an organization (taken from the sample database created by the DB2® software installation) and the other representing medical records of patients (taken from "Avitek Medical Records Development Tutorials" by BEA Systems, Inc.[12]). Both were run using our own PII model. The algorithm was able to discover all columns that contained data from the classes in the PII model. For example, a person's first name was discovered in both databases under numerous forms such as FIRSTNAME, FIRST_NAME and FIRSTNME. Date of birth was found both as DOB and as BIRTHDATE. In the first round of the implementation, a few false positives were found. For example, a column called REGION matched with the class Religion. However after correcting the division into syllables for the word religion, this phenomenon was eliminated. This correction was done in an external resource file and not in the code itself.

We compared the results of running the discovery algorithm with different components of the search algorithm disabled. First we ran the algorithm without the use of additional synonyms for each field name. Then we ran it with synonyms but without the syllabifying techniques described in Section 4.2. Finally we ran it with the syllabifying techniques but without verifying the type of the column. Columns that fit a certain category in the input model but that were not discovered were considered false negatives, and columns that were discovered but did not actually match the category were considered false positives. A summary of these results is shown in Table 1.

The conclusion from our work is that the main advantages of our approach are the combination of different matching techniques and the extraction of most of the search logic to external files that can be easily changed, as opposed to changing the actual searching algorithms. In addition we suggest an application

[12] `http://download.oracle.com/docs/cd/E13222_01/wls/docs100/medrec_tutorials/index.html`

Algorithm	Total no. of hits	No. of false negatives	No. of false positives
Original alg.	37	0	0
No synonyms	35	2	0
No syllabifying	29	8	0
No type matching	41	0	14

Table 1. The results of running the classification using different algorithms

model in which there is an initial iterative "learning phase" on some sample databases in which the models are manually tweaked to optimize the discovery results. Using our NeOn-based tool, an administrator can easily navigate between the discovery results, the original models and the database representations, thus validating the results and refining the models.

5.3 Summary

We presented a model-based approach to data classification and demonstrated its advantages using a reference implementation for relational databases. We showed that this is a flexible solution that can be used to automatically classify information from different data types and sources, in a fine-grained manner, with no dependency on the existing or future policies that will be applied to the discovered data.

The model-based approach provides extensibility and flexibility in that the ontologies can be easily extended, edited or replaced without the need to change the discovery algorithm itself. The same ontology can be used as input to different classifiers that classify different types of data sources. In addition, all classification results are saved in the same format, providing a unified representation of all classified data sources and enabling uniform policy expression and enforcement across the enterprise.

Using RDF to represent both the ontologies and the results maximizes modularity and extensibility and facilitates easy navigation between the results, the models and the data sources. The ontology can serve as a centralized point to manage all valuable information in the organization and enables easy location of all related pieces of data in one click. The RDF representation of the classification results enables high connectivity between the search results on the different data sources, and increases the variety of possible uses for these results, also enabling the application of semantic queries.

6 Conclusions and Future work

There are many axes in which this work can be extended. One such axis is the variety of data types that can be classified. In the implementation presented above we classified relational (structured) data; doing so on unstructured or

semi-structured data is not trivial because we would like to be able to link a certain class to a specific data element in the document and not to the entire document.

An additional axis is the development and combination of the ontologies themselves. We developed a PII model as the core knowledge for privacy purposes. However this is just an example. For different use cases in the various industries, additional models will be required, each with its specific domain knowledge and expertise.

Another direction is taking advantage of the RDF representation of the results to apply reasoning and additional Semantic Web technologies for policy enforcement, semantic data queries and more. There is a very rich set of existing and developing tools that can be applied to gain new insight into the organization's data and enable getting one step closer to the vision of the Semantic Enterprise.

One more aspect that can be investigated is improving the discovery algorithm itself, using additional available information.

References

1. Beliefnetworks - Semantically Secure Unstructured Data Cassification. White paper (2008), `http://www.beliefnetworks.net/docs/classunstructdata.pdf`
2. Autonomy - Meaning-Based Computing Technology. Press release (2009), `http://www.autonomy.com/content/News/Releases/2009/0817.en.html`
3. Ben-Dor, A., Friedman, N., Yakhini, Z.: Class discovery in gene expression data. Annual Conference on Research in Computational Molecular Biology pp. 31–38 (2001)
4. Bizer, C., Cyganiak, R.: D2RQ V0.2 - Treating Non-RDF Relational Databases as Virtual RDF Graphs. Tech. rep., School of Business & Economics at the Freie Universit?t Berlin (2004)
5. Boley, H., Forschungszentrum, D., Gmbh, K.I.: Relationships between logic programming and RDF. In: 14th Workshop Logische Programmierung (2000)
6. Butler, M., Reynolds, D., Dickinson, I., McBride, B., Grosvenor, D., Seaborne, A.: Semantic Middleware for E-Discovery. In: IEEE International Conference on Semantic Computing (2009)
7. Carroll, J.J., Dickinson, I., Dollin, C., Seaborne, D.R.A., Wilkinson, K.: Jena: Implementing the Semantic Web Recommendations. Tech. rep., HP Laboratories (2003)
8. Castells, P., Foncillas, B., Lara, R., Rico, M., Alonso, J.L.: Semantic Web Technologies for Economic and Financial Information Management. In: The Semantic Web: Research and Applications. vol. 3053, pp. 473–487 (2004)
9. Holger, P.H., Studer, L.R., Tran, T.: The NeOn Ontology Engineering Toolkit. In: ISWC (2009)
10. Jenkins, C., Jackson, M., Burden, P., Wallis, J.: Automatic RDF Metadata Generation for Resource Discovery. The International Journal of Computer and Telecommunications Networking 32, 1305 – 1320 (1999)
11. Kagal, L.: Rei: A Policy Language for the Me-Centric Project. Tech. rep., HP Labs (2002)

12. Kagal, L., Hanson, C., Weitzner, D.J.: Using Dependency Tracking to Provide Explanations for Policy Management. In: 2008 IEEE Workshop on Policies for Distributed Systems and Networks. pp. 54–61 (2008)
13. Krishnamurthy, R., Li, Y., Raghavan, S., Reiss, F., Vaithyanathan, S., Zhu, H.: SystemT: a system for declarative information extraction. ACM SIGMOD Record 37, 7–13 (2008)
14. Kruk, S.R.: FOAF-Realm - control your friends' access to the resource. In: FOAF Workshop proceedings (2004)
15. de Laborda, C.P., Conrad, S.: RelationalOWL: a data and schema representation format based on OWL. In: Conferences in Research and Practice in Information Technology. pp. 89–96 (2005)
16. Langegger, A., W?ß, W., Bl?chl, M.: A Semantic Web Middleware for Virtual Data Integration on the Web. In: The Semantic Web: Research and Applications, pp. 493–507. Springer Berlin/Heidelberg (2008)
17. Maedche, A., Motik, B., Stojanovic, L., Studer, R., Volz, R.: Ontologies for enterprise knowledge management. IEEE Intelligent Systems 18, 26–33 (2003)
18. Miller, D.J.: A Mixture Model and EM-Based Algorithm for Class Discovery, Robust Classification, and Outlier Rejection in Mixed Labeled/Unlabeled Data Sets. IEEE Transactions on Pattern Analysis and Machine Intelligence (2003)
19. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C recommendation, W3C (2008)
20. Seaborne, A.: RDQL query language for RDF. W3C member submission, W3C (2004)
21. Song, Y., Zhou, D., Huang, J., Zha, I.G.C.Z., Giles, C.L.: Boosting the feature space: Text classification for unstructured data on the web. In: ICDM. pp. 1064–1069 (2006)
22. Sperberg-McQueen, C.M., Miller, E.: On mapping from colloquial XML to RDF using XSLT. In: Extreme Markup Languages (2004)
23. Staab, S., Studer, R., Schnurr, H.P., Sure, Y.: Knowledge Processes and Ontologies. IEEE Intelligent Systems 16, 26–34 (2001)
24. Taghva, K., Borsack, J., Coombs, J., Condit, A., Lumos, S., Nartker, T.: Ontology-based Classification of Email. In: International Conference on Information Technology: Computers and Communications. p. 194 (2003)
25. Tim Berners-Lee, Dan Connolly, L.K.Y.S.J.A.H.: N3Logic: A logical framework for the World Wide Web. TPLP 8(3), 249–269 (2008)
26. Uszok, A., Bradshaw, J.M., Johnson, M., Jeffers, R., Tate, A., Dalton, J., Aitken, S.: KAoS Policy Management for Semantic Web Services. IEEE Intelligent Systems 19, 32–41 (2004)
27. Warren, P.W., Davies, N.J.: Managing the risks from information - through semantic information management. BT Technology Journal 25, 178–191 (2007)
28. Weitzner, D.J., Abelson, H., Berners-lee, T., Hanson, C., Hendler, J., Kagal, L., Mcguinness, D.L., Sussman, G.J., Waterman, K.K.: Transparent accountable data mining: New strategies for privacy protection. Tech. rep., MIT-CSAIL (2006)
29. Xiaoyue, W., Rujiang, B.: Applying RDF Ontologies to Improve Text Classification. In: International Conference on Computational Intelligence and Natural Computing. vol. 2, pp. 118–121 (2009)